Generating A Parsing Lexicon from an LCS-Based Lexicon

Necip Fazıl Ayan and Bonnie J. Dorr

Department of Computer Science
University of Maryland
College Park, 20742, USA
{nfa, bonnie}@umiacs.umd.edu

Abstract

This paper describes a technique for generating parsing lexicons for a principle-based parser (Minipar). Our approach maps lexical entries in a large LCS-based repository of semantically classified verbs to their corresponding syntactic patterns. A by-product of this mapping is a lexicon that is directly usable in the Minipar system. We evaluate the accuracy and coverage of this lexicon using LDOCE syntactic codes as a gold standard. We show that this lexicon is comparable to the hand-generated Minipar lexicon (i.e., similar recall and precision values). In a later experiment, we automate the process of mapping between the LCS-based repository and syntactic patterns. The advantage of automating the process is that the same technique can be applied directly to lexicons we have for other languages, for example, Arabic, Chinese, and Spanish.

1. Introduction

This paper describes a technique for generating parsing lexicons for a principle-based parser (Minipar (Lin, 1993; Lin, 1998)) using a lexicon that is semantically organized according to Lexical-Conceptual Structure (LCS) (Dorr, 1993; Dorr, 2001)—an extended version of the verb classification system proposed by (Levin, 1993).[1] We aim to determine how much syntactic information we can obtain from this resource, which extends Levin's original classification as follows: (1) it contains 50% more verbs and twice as many verb entries (Dorr, 1997)—including new classes to accommodate previously unhandled verbs and phenomena (e.g., clausal complements); (2) it incorporates theta-roles which, in turn, are associated with a thematic hierarchy for generation (Habash and Dorr, 2001); and (3) it provides a higher degree of granularity, i.e., verb classes are sub-divided according to their aspectual characteristics (Olsen et al., 1997).

More specifically, we provide a general technique for projecting this broader-scale semantic (language-independent) lexicon onto syntactic entries, with the ultimate objective of testing the effects of such a lexicon on parser performance. Each verb in our semantic lexicon is associated with a class, an LCS representation, and a thematic grid.[2] These are mapped systematically into syntactic representations. A by-product of this mapping is a lexicon that is directly usable in the Minipar system.

Several recent lexical-acquisition approaches have produced new resources that are ultimately useful for syntactic analysis. The approach that is most relevant to ours is that of (Stevenson and Merlo, 2002b; Stevenson and Merlo, 2002a), which involves the derivation of verb classes from syntactic features in corpora. Because their approach is unsupervised, it provides the basis for automatic verb classification for languages not yet seen. This work is instrumental in providing the basis for wide-spread applicability of our technique (mapping verb classes to a syntactic parsing lexicon), as verb classifications become increasingly available for new languages over the next several years.

An earlier approach to lexical acquisition is that of (Grishman et al., 1994), an effort resulting in a large resource called Comlex—a repository containing 38K English headwords associated with detailed syntactic patterns. Other researchers (Briscoe and Carroll, 1997; Manning, 1993) have also produce subcategorization patterns from corpora. In each of these cases, data collection is achieved by means of statistical ex-

[1] We focus only on verb entries as they are cross-linguistically the most highly correlated with lexical-semantic divergences.

[2] Although Lexical Conceptual Structure (LCS) is the primary semantic representation used in our verb lexicon, it is not described in detail here (but see (Dorr, 1993; Dorr, 2001). For the purpose of this paper, we rely primarily on the thematic grid representation, which is derived from the LCS. Still we refer to the lexicon as "LCS-based" as we store all of these components together in one large repository: http://www.umiacs.umd.edu/~bonnie/LCS_Database_Documentation.html.

traction from corpora; there is no semantic basis and neither is intended to be used for multiple languages.

The approaches of (Carroll and Grover, 1989) and (Egedi and Martin, 1994) involve acquisition English lexicons from entries in LDOCE and *Oxford Advanced Learner's Dictionary* (OALD), respectively. The work of (Brent, 1993) produces a lexicon from a grammar—the reverse of what we aim to do. All of these approaches are specific to English. By contrast, our goal is to have a unified repository that is transferable to other languages—and from which our parsing (and ultimately generation) grammars may be derived.

For evaluation purposes, we developed a mapping from the codes of Longman's Dictionary of Contemporary English (LDOCE (Procter, 1978))—the most comprehensive online dictionary for syntactic categorization—to a set of syntactic patterns. We use these patterns as our gold standard and show that our derived lexicon is comparable to the hand-generated Minipar lexicon (i.e., similar recall and precision values). In a later experiment, we automate the process of mapping between the LCS-based repository and syntactic patterns—with the goal of portability: We currently have LCS lexicons for English, Arabic, Spanish, and Chinese, so our automated approach allows us to produce syntactic lexicons for parsing in each of these languages.

Section 2. presents a brief description of each code set we use in our experiments. In Section 3., we explain how we generated syntactic patterns from three different lexicons. In Section 4., we discuss our experiments and the results. Section 5. describes ongoing work on automating the mapping between LCS-based representations and syntactic patterns. Finally, we discuss our results and some possible future directions.

2. Code Descriptions

In many online dictionaries, verbs are classified according to the arguments and modifiers that can follow them. Most dictionaries use specific codes to identify transitivity, intransitivity, and ditransitivity. These broad categories may be further refined, e.g., to distinguish verbs with NP arguments from those with clausal arguments. The degree of refinement varies widely.

In the following subsections, we will present three different code sets. As shown in Figure 1, the first of these (OALD) serves as a mediating representation in the mapping between Minipar codes and syntactic patterns. The LCS lexicon and LDOCE codes are mapped directly into syntactic patterns, without an intervening representation. The patterns resulting from the LDOCE are taken as the gold standard, serving as the basis of comparison between the Minipar- and LCS-based lexicons.

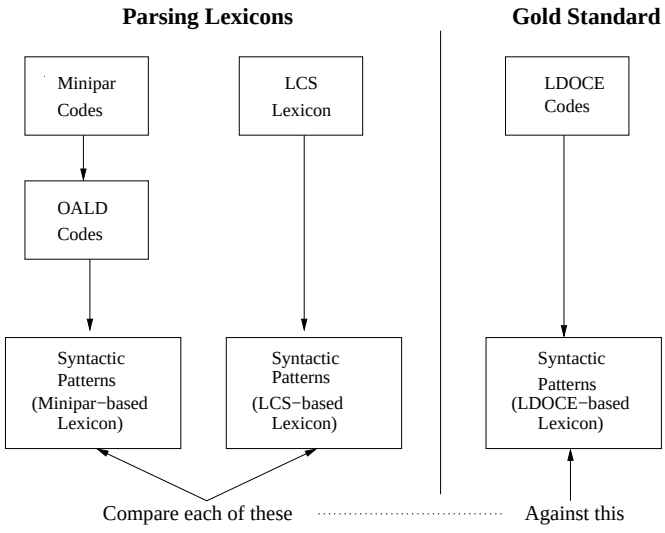

Figure 1: A Comparison between Minipar- and LCS-based Lexicons using LDOCE as the Gold Standard

2.1. OALD Codes

This code set is used in Oxford Advanced Learner's Dictionary, a.k.a OALD (Mitten, 1992). The verbs are categorized into 5 main groups: Intransitive verbs, transitive verbs, ditransitive verbs, complex transitive verbs, and linking verbs. Each code is of the form $Sa_1[.a_2]$ where S is the first letter of the verb categorization ($S \in \{I, T, D, C, L\}$ for the corresponding groups), and $a_1, a_2, \ldots$ are the argument types. If a code contains more than one argument, each argument is listed serially. Possible argument types are n for nouns, f for finite clauses (*that* clauses), g for "-ing" clauses, t for infinitive clauses, w for finite clauses beginning with "-wh", i for bare infinitive clauses, a for adjective phrases, p for prepositions and pr for prepositional phrases.

For example, Tn refers to the verbs followed by a noun ('She read the book'), $Tn.pr$ refers to the verbs followed by a noun and a prepositional phrase ('He opened the door with a latch'), and $Dn.n$ refers to the verbs followed by two nouns ('She taught the children French'). The number of codes in OALD code set is 32 and the codes are listed in Table 1.

OALD codes are simplistic in that they do not include modifiers. In addition, they also do not explicitly specify which prepositions can be used in the PPs.

2.2. Minipar Codes

The Minipar coding scheme is an adaptation of the OALD codes. Minipar extends OALD codes by pro-

Categorization	OALD Codes
Intransitive verbs	{I, Ip, Ipr, In/pr, It}
Transitive verbs	{Tn, Tn.pr, Tn.p, Tf, Tw, Tt, Tg, Tn.t, Tn.g, Tn.i}
Complex Transitive verbs	{Cn.a, Cn.n, Cn.n/a, Cn.t, Cn.g, Cn.i}
Ditransitive verbs	{Dn.n, Dn.pr, Dn.f, Dn.t, Dn.w, Dpr.f, Dpr.w, Dpr.t}
Linking verbs	{La, Ln}

Table 1: OALD Code Set: The Basis of Minipar Codes

viding a facility for specifying prepositions, but only 8 verbs are encoded with these prepositional codes in the official Minipar distribution. In these cases, the codes containing pr are refined to be $pr.prep$, where $prep$ is the head of the PP argument.[3] In addition, Minipar codes are refined in the following ways:

1. Optional arguments are allowed, e.g., $T[n].pr$ describes verbs followed by an optional noun and a PP. This is equivalent to the combination of the OALD codes $Tn.pr$ and Ipr.

2. Two or more codes may be combined, e.g., Tfgt describes verbs followed by a clause that is finite, infinitive, or gerundive ("-ing").

3. Prepositions may be specified in prepositional phrases. Some of the codes containing pr as an argument are converted into $pr.prep$ in order to declare that the prepositional phrase can begin with only the specified preposition $prep$.

The set of Minipar codes contain 66 items. We will not list them here since they are very similar to the ones in Table 1, with the modifications described above.

2.3. LDOCE Codes

LDOCE has a more detailed code set than that of OALD (and hence Minipar). The codes include both arguments and modifiers. Moreover, prepositions are richly specified throughout the lexicon. The syntax of the codes is either *CN* or *CN-Prep*, where *C* corresponds to the verb sub-categorization (as in the generic OALD codes) and N is a number, which corresponds to different sets of arguments that can follow the verb. For example, T1-ON refers to verbs that are followed by a noun and a PP with the head *on*. The number of codes included in this set is 179. The meaning of each is described in Table 2.

3. Our Approach

Our goal is evaluate the accuracy and coverage of a parsing lexicon where each verb is classified according to the arguments it takes. We use syntactic patterns as the basis of the comparison between our parsing lexicon and the original lexicon used in Minipar.

[3]This extension is used only for the preposition *as* for the verbs *absolve, accept, acclaim, brand, designate, disguise, fancy,* and *reckon*.

Number	Arguments
1	one or more nouns
2	bare infinitive clause
3	infinitive clause
4	-ing form
5	-that clause
6	clauses with a wh- word
7	adjective
8	past participle
9	descriptive word or phrase

Table 2: LDOCE Number Description

Syntactic patterns simply list the type of the arguments one by one, including the subject. Formally, a syntactic pattern is $a_1, a_2, \ldots$ where a_i is an element of NP, AP, PP, FIN, INF, BARE, ING, WH, PREP, corresponding to noun phrases, adjective phrases, prepositional phrases, clauses beginning with "that", infinitive clauses, bare infinitive clauses, "-ing" clauses, "-wh" clauses and prepositions, respectively. Prepositional phrases may be made more specific by including the heads, which is done by PP.$prep$ where $prep$ is the head of the prepositional phrase. The first item in the syntactic pattern gives the type of the subject.

Our initial attempts at comparing the Minipar- and LCS-based lexicons involved the use of the OALD code set instead of syntactic patterns. This approach has two problems, which are closely related. First, using the class number and thematic grids as the basis of mapping from the LCS lexicon to OALD codes is a difficult task because of the high degree of ambiguity. For example, it is hard to choose among four OALD codes (Ln, La, Tn or Ia) for the thematic grid _th_pred, regardless of the Levin class. In general, the grid-to-OALD mapping is so ambiguous that maintaining consistency over the whole LCS lexicon is virtually impossible.

Secondly, even if we are able to find the correct OALD codes, it is not worth the effort because all that is needed for the parsing lexicon is the type and number of arguments that can follow the verb. For example, $Cn.n$ (as in *"appoint him king"*) and $Dn.n$ (as in *"give him a book"*) both correspond to two

NPs, but the second NP is a direct object in the former case and an indirect object in the latter. Since the parser relies ultimately on syntactic patterns, not codes, we can eliminate this redundancy by mapping any verb in either of these two categories directly into the [NP.NP.NP] pattern. Thus, using syntactic patterns is sufficient for our purposes.

Our experiments revealed additional flexibility in using syntactic patterns. Unlike the OALD codes (which contain at most two arguments or modifiers), the thematic grids consist of up to 4 modifiers. Mapping onto syntactic patterns instead of onto OALD codes allows us to use all arguments in the thematic grids. For example, [NP.NP.PP.from.PP.to] is an example of transitive verb with two prepositional phrases, one beginning with *from* and the other beginning with *to*, as in "*She drove the kids from home to school.*"

In the following subsections, we will examine the mapping into these syntactic patterns from: (1) the LCS lexicon; (2) the Minipar codes; and (3) the LDOCE codes.

3.1. Mapping from the LCS Lexicon to Syntactic Patterns

The LCS lexicon consists of verbs grouped into classes based on an adapted version of verb classes (Levin, 1993) along with the thematic grid representations (see (Dorr, 1993; Dorr, 2001)). We automatically assigned syntactic patterns for each verb in the LCS lexicon using its semantic class number and thematic grid. The syntactic patterns we used in our mapping specify prepositions for entries that require them. For example, the grid _ag_th_instr(with) is mapped onto [NP.NP.PP.with] instead of a generic pattern [NP.NP.PP].

More generally, thematic grids contain a list of arguments and modifiers, and they can be obligatory (indicated by an underscore before the role) or optional(indicated by a comma before the role). The arguments can be one of AG, EXP, TH, SRC, GOAL, INFO, PERC, PRED, LOC, POSS, TIME, and PROP. The logical modifiers can be one of MOD-POSS, BEN, INSTR, PURP, MOD-LOC, MANNER, MOD-PRED, MOD-PERC and MOD-PROP. If the argument or the modifier is followed by parenthesis, the corresponding element is a prepositional phrase and its head must be the one specified between the parentheses (if there is nothing between parentheses, PP can begin with any preposition).

Our purpose is to find the set of syntactic patterns for each verb in LCS lexicon using its Levin class and thematic grid. Since each verb can be in many classes and we aim at assigning syntactic patterns based on the semantic classes and thematic grids, there are three possible mapping methodologies:

1. Assign one or more patterns to each class.
2. Assign one or more patterns to each thematic grid.
3. Assign one or more patterns to each pair of class and thematic grid.

The first methodology fails for some classes because the distribution of syntactic patterns over a specific class is not uniform. In other words, attempting to assign only a set of patterns to each class introduces errors because some classes are associated with more than one syntactic frame. For example, class 51.1.d includes three thematic grids: (1) _th,src; (2) _th,src(from); and (3) _th,src(),goal(). We can either assign all patterns for all of these thematic grids to this class or we can choose the most common one. However, both of these approaches introduce errors: The first will generate redundant patterns and the second will assign incorrect patterns to some verbs. (This occurs because, within a class, thematic grids may vary with respect to their optional arguments or the prepositional head associated with arguments or modifiers.)

The second methodology also fails to provide an appropriate mapping. The problem is that some thematic grids correspond to different syntactic patterns in different classes. For example, the thematic grid _th_prop corresponds to 3 different syntactic patterns: (1) [NP.NP] in class 024 and 55.2.a; (2) [NP.ING] in classes 066, 52.b, and 55.2.b; and (3) [NP.INF] in class 005. Although the thematic grid is the same in all of these classes, the syntactic patterns are different.

The final methodology circumvents the two issues presented above (i.e., more than one grid per class and more than one syntactic frame per thematic grid) as follows: If a thematic grid contains an optional argument, we create two mappings for that grid, one in which the optional argument is treated as if it were not there and one in which the argument is obligatory. For example, _ag_th,goal() is mapped onto two patterns [NP.NP] and [NP.NP.PP]. If the number of optional arguments is X, then the maximum number of syntactic patterns for that grid is 2^X (or perhaps smaller than 2^X since some of the patterns may be identical).

Using this methodology, we found the correct mapping for each class and thematic grid pair by examining the verbs in that class and considering all possible syntactic patterns for that pair. This is a many-to-many mapping, i.e. one pattern can be used for different

OALD Code	Syntactic Patterns
I	[NP]
Tn	[NP.NP]
T[n].pr	[NP.NP] and [NP.NP.PP]
Cn.a	[NP.NP.AP]
Cn.n	[NP.NP.NP]
Cn.n/a	[NP.NP.PP.as]
Cn.i	[NP.NP.BARE]
Dn.n	[NP.NP.NP]

Table 3: Mapping From OALD to Syntactic Patterns

LDOCE Code	Syntactic Patterns
I-ABOUT	[NP.PP.about]
I2	[NP.BARE]
L9-WITH	[NP.PP.with]
T1	[NP.NP]
T5	[NP.FIN]
D1	[NP.NP.NP]
D3	[NP.NP.INF]
V4	[NP.NP.ING]

Table 4: Mapping From LDOCE to Syntactic Patterns

pairs and each pair may be associated with more than one pattern. Each verb in each class is assigned the corresponding syntactic patterns according to its thematic grid. Finally, for each verb, we combined all patterns in all classes containing this particular verb in order to generate the lexicon. We will refer to the resulting lexicon as the LCS-based lexicon in Section 4..

3.2. Mapping from Minipar Codes To Syntactic Patterns

Minipar codes are converted straightforwardly into syntactic patterns using the code specification in (Mitten, 1992). An excerpt of the mapping is given in Table 3. This mapping is one-to-many as exemplified by the code $T[n].pr$. Moreover, the set of syntactic patterns extracted from Minipar does not include some patterns such as [NP.PP] (and related patterns) because Minipar does not include modifiers in its code set.

As a result of this mapping, we produced a new lexicon from Minipar entries, where each verb is listed along with the set of syntactic patterns. We will refer to this lexicon as the Minipar-based lexicon in Section 4..

3.3. Mapping from LDOCE Codes to Syntactic Patterns

Similar to the mapping from Minipar to the syntactic patterns, we converted LDOCE codes to syntactic patterns using the code specification in (Procter, 1978). An excerpt of the mapping is given in Table 4.

Each LDOCE code was mapped manually to one or more patterns. LDOCE codes are more refined than the generic OALD codes, but mapping each to syntactic patterns provides an equivalent mediating representation for comparison. For example, LDOCE codes D1-AT and T1-AT are mapped onto [NP.NP.PP.at] by our mapping technique. Again, this is a many-to-many mapping but only a small set of LDOCE codes map to more than one syntactic pattern.

As a result of this mapping, we produced a new lexicon from LDOCE entries, similar to Minipar lexicon. We will refer to this lexicon as the LDOCE-based lexicon in Section 4..

4. Experiments and Results

To measure the effectiveness of our mapping from LCS entries to syntactic patterns, we compared the precision and recall our derived LCS-based syntactic patterns with the precision and recall of Minipar-based syntactic patterns, using LDOCE-based syntactic patterns as our "gold standard".

Each of the three lexicons contains verbs along with their associated syntactic patterns. For experimental purposes, we convert these into pairs. Formally, if a verb v is listed with the patterns $p_1, p_2, \ldots$, we create pairs (v, p_1), (v, p_2) and so on. In addition, we have made the following adjustments to the lexicons, where L is the lexicon under consideration (Minipar or LCS):

1. Given that the number of verbs in each of the two lexicons is different and that neither one completely covers the other, we take only those verbs that occur in both L and LDOCE, for each L, while measuring precision and recall.

2. In the LDOCE- and Minipar-based lexicons, the number of arguments is never greater than 2. Thus, for a fair comparison, we converted the LCS-based lexicon into the same format. For this purpose, we simply omit the arguments after the second one if the pattern contains more than two arguments/modifiers.

3. The prepositions are not specified in Minipar-based lexicon. Thus, we ignore the heads of the prepositions in LCS-based lexicon, i.e., if the pattern includes [PP.$prep$] we take it as a [PP].

Precision and recall are based on the following inputs:

A = Number of pairs in L occurring in LDOCE
B = Number of pairs in L NOT occurring in LDOCE
C = Number of pairs in LDOCE NOT occurring in L

That is, given a syntactic pattern encoded lexicon L, we compute:

(1) The *precision* of $L = \frac{A}{A+B}$;
(2) The *recall* of $L = \frac{A}{A+C}$.

Verbs in LDOCE Lexicon	5648
Verbs in LCS Lexicon	4324
Common verbs in LCS and LDOCE	3813
Pairs in LCS Lexicon	9510
Pairs in LDOCE Lexicon	9342
Pairs in in LCS and LDOCE	5787
Verbs fetched completely	1826
Precision	61%
Recall	62%

Table 5: Experiment on LCS-based Lexicon

	All Verbs in Minipar Lexicon	**Common verbs with LCS Lexicon**
Verbs in LDOCE Lexicon	5648	5648
Verbs in Minipar Lexicon	9478	4044
Common verbs in Minipar and LDOCE	5385	3763
Pairs in Minipar Lexicon	9006	6835
Pairs in LDOCE Lexicon	11726	9260
Pairs in Minipar and LDOCE	7229	5543
Verbs fetched completely	2755	1719
Precision	80%	81%
Recall	62%	60%

Table 6: Experiments on Minipar-based Lexicon

We compare two results: one where L is the Minipar-based lexicon and one where L is the LCS-based lexicon. Table 5 gives the number of verbs used in the LCS-based lexicon and the LDOCE-based lexicon, showing the precision and recall. The row showing the number of verbs fetched completely gives the number of verbs in the LCS lexicon which contains all the patterns in the LDOCE entry for the same verb. The precision is 61% and the recall is 62%.

We did the same experiment for the Minipar-based lexicon in two different ways, first with all the verbs in the Minipar lexicon and then with only the verbs occurring in both the LCS and Minipar lexicons. The second approach is useful for a direct comparison between the Minipar- and LCS-based lexicons. As before, we used the LDOCE-based lexicon as our gold standard. The results are shown in Table 6. The definitions of entries are the same as in Table 5.

The number of Minipar verbs in Minipar occurring in the LCS lexicon is different from the total number of LCS verbs because some LCS verbs (280 of them) do not appear in Minipar lexicon. The results indicate that the Minipar-based lexicon yields much better precision, a 20% difference with respect to the LCS-based lexicon. The recall is low because Minipar does not take modifiers into account most of the time. This results in missing nearly all patterns with PPs, such as [NP.PP] and [NP.NP.PP]. The recall achieved is nearly same as that of the LCS-based lexicon, only 2% lower.

Verbs in LDOCE Lexicon	5648
Verbs in Intersection Lexicon	3671
Common verbs in Int. and LDOCE	3415
Pairs in Intersection Lexicon	4489
Pairs in LDOCE Lexicon	8465
Pairs in Int. and LDOCE	4088
Verbs fetched completely	1265
Precision	91%
Recall	48%

Table 7: Experiment on Intersection Lexicon

Finally, we conducted an experiment to see how the intersection of the Minipar and LCS lexicons compares to the LDOCE-based lexicon. For this experiment, we included only the verbs and patterns occurring in both lexicons. The results are shown in Table 7 in a format similar to previous tables.

The number of common verbs differs from the previous ones because we omit the verbs which do not have any patterns across the two lexicons. The results are not surprising: High precision is achieved because only those patterns that occur in both lexicons are included in the intersection lexicon; thus, the total number of pairs is reduced significantly. For the same reason, the recall is significantly reduced.

The highest precision is achieved by the intersection of two lexicons, but at the expense of recall. We found that the precision was higher for Minipar than for the LCS lexicon, but when we examined this in more detail, we found that this was almost entirely due to "double counting" of entries with optional modifiers in the LCS-based lexicon. For example, the single LCS-based grid _ag_th,instr(with) corresponds to two syntactic patterns, [NP.NP] and [NP.NP.PP], while LDOCE views these as the single pattern [NP.NP]. Specifically, 53% of the non-matching LCS-based patterns are [NP.NP.PP]—and 93% of these co-occur with [NP.NP]. Similarly, 13% of the non-matching LCS-based patterns are pattern [NP.PP]—and 80% of these co-occur with [NP].

This is a significant finding, as it reveals that our precision is spuriously low in our comparison with the "gold standard." In effect, we should be counting the LCS-based pattern [NP.NP.PP]/[NP.NP] to be a match against the LDOCE-based pattern [NP.NP]—which is a fairer comparison since neither LDOCE nor Minipar takes modifiers into account. (We henceforth refer to LCS-based the co-occurring patterns [NP.NP.PP]/[NP.NP] and [NP.PP]/[NP] as overlapping pairs.) To observe the degree of the impact of optional

	Minipar Lexicon (All verbs in Minipar Lexicon)	Minipar Lexicon (Common verbs with LCS Lexicon)	LCS Lexicon	Intersection of Minipar and LCS Lexicons
Precision	80%	81%	61%	91%
Enhanced Precision	80%	81%	**80%**	91%
Recall	62%	60%	62%	48%

Table 8: Precision and Recall Summary: Minipar- and LCS-based Lexicons

modifiers, we computed another precision value for the LCS-based lexicon by counting overlapping patterns once instead of twice. With this methodology, we achieved 80% (enhanced) precision. This precision value is nearly same as the value achieved with the current Minipar lexicon. Table 8 summarizes all results in terms of precision and recall.

The enhanced precision is an important and accurate indicator of the effectiveness of our approach, given that overlapping patterns arise because of (optional) modifiers. When we ignore those modifiers during our mapping process, we achieve nearly the same precision and recall with the current Minipar lexicon, which also ignores the modifiers in its code set. Moreover, overlapping patterns in our LCS-based lexicon do not affect the performance of the parser, other than to induce a more sophisticated handling of modifiers (which presumably would increase the precision numbers, if we had access to a "gold standard" that includes modifiers). For example, Minipar attaches modifiers at the clausal level instead of at the verbal level even in cases where the modifier is obviously verbal—as it would be in the LCS-based version of the parse in the sentence *She rolled the dough [PP into cookie shapes]*.

5. Ongoing Work: Automatic Generation of Syntactic Patterns

The lexicon derived from the hand-crafted mapping between the LCS lexicon and the syntactic patterns is comparable to the current Minipar lexicon. However, the mapping required a great deal of human effort, since each semantic verb class must be examined by hand in order to identify appropriate syntactic patterns. The process is error-prone, laborious, and time-intensive (approximately 3-4 person-months). Moreover, it requires that the mapping be done again by a human every time the LCS lexicon updated.

In a recent experiment, we developed an automated mapping (in 2 person-weeks) that takes into account both semantic roles and some additional features stored in the LCS database, without reference to the class number. The mapping is based primarily on the thematic role, however in some situations the thematic roles themselves are not sufficient to determine the type of the argument. In such cases, the correct form is assigned using featural information associated with that specific verb in the LCS database.

Table 10 summarizes the automated mapping rules. The thematic role "prop" is an example of a case where featural information is necessary (e.g., (cform inf)), as there are five different patterns to choose from for this thematic role. Similarly, whether a "pred" role is an NP or AP is determined by featural information. For example, this role becomes an AP for the verb $behave$ in class 29.6.a while it is mapped onto an NP for the verb $carry$ in class 54.2. In the cases where the syntactic pattern is ambiguous and there is no specification for the verbs, default values are used for the mapping: BARE for "prop", AP for "pred" and NP for "perc".

Syntactic patterns for each thematic grid are computed by combining the results of the mapping from each thematic role in the grid to a syntactic pattern, one after another. If the grid includes optional roles, every possibility is explored and the syntactic patterns for each of them is included in the whole list of patterns for that grid. For example, the syntactic patterns for _ag_th,instr(with) include the patterns for both _ag_th and _ag_th_instr(with), which are [NP.NP] and [NP.NP.PP.with].

Note that this approach eliminates the need for using the same syntactic patterns for all verbs in a specific class: Verbs in the same class can be assigned different syntactic patterns with the help of additional features in the database. Thus, we need not rely on the semantic class number at all during this mapping. We can easily update the resulting lexicons when there is any change on the semantic classes or thematic grids of some verbs.

This experiment resulted in a parsing lexicon that has virtually the same precision/recall as that of the manually generated LCS-based lexicon above—with the exception of a 1% difference in recall.[4] (See Ta-

[4]The difference in recall is only slightly lower—61.36% for the automated mapping, compared to 61.95% for the hand-generated mapping—because the hand-generated

Verbs in LDOCE Lexicon	5648
Verbs in LCS Lexicon	4162
Common verbs in LCS and LDOCE	3661
Pairs in LCS Lexicon	9084
Pairs in LDOCE Lexicon	9004
Pairs in LCS and LDOCE	5523
Verbs fetched completely	1747
Precision	61%
Enhanced Precision	80%
Recall	61%

Table 9: Precision and Recall of Automatic Generation of Syntactic Patterns

Thematic Role	**Syntactic Patterns**
particle	PREP
prop(...), mod-prop(...), info(...)	FIN or INF or ING or PP
all other role(...)	PP
th, exp, info	FIN or INF or ING or NP
prop	NP or ING or INF or FIN or BARE
pred	AP or NP
perc	[NP.ING] or [NP.BARE]
all other roles	NP

Table 10: Syntactic Patterns Corresponding to Thematic Roles

ble 9.) As in the case of the manually generated mappings, the enhanced precision for automatically generated mappings is 80%, which is only 1% lower than that of the Minipar-based lexicon.

Our approach demonstrates that examination of thematic-role and featural information in the LCS-based lexicon is sufficient for executing this mapping automatically. Automating our approach gives us the flexibility of re-running the program if the structure of the database changes (e.g., an LCS representation is modified or class membership changes) and of porting to a new language with minimal effort.

6. Discussion

In all experiments reported above, both the LCS- and Minipar-based lexicons yield low recall values. Upon further investigation, we found that LDOCE is too specific in assigning codes to verbs. Most of the patterns associated with the verbs are rare—cases not considered in the LCS- and Minipar-based lexicons. Because of that, we believe that the recall values will improve if we take only a subset of LDOCE-based lexicon, e.g., those associated with the most frequent verb-pattern pairs in a large corpus. This is a future research direction considered in the next section.

The knowledgeable reader may question the mapping of a Levin-style lexicon into syntactic codes, given that Levin's original proposal is to investigate verb meaning through examination of syntactic patterns, or *alternations*, in the first place. As alluded to in Section 1., there are several ways in which this database has become more than just a "semantified" version of a syntactic framework; we elaborate on this further here.

Levin's original framework omitted a large number of verbs—and verb senses for existing Levin verbs—which we added to the database by semi-automatic techniques. Her original framework contained 3024 verbs in 192 classes numbering between 9.1 and 57—a total of 4186 verb entries. These were grouped together primarily by means of syntactic alternations. Our augmented database contains 4432 verbs in 492 classes with more specific numbering (e.g., "51.3.2.a.ii") including additional class numbers for new classes that Levin did not include in her work (between 000 and 026)—a total of 9844 verb entries. These were categorized according to semantic information (using WordNet synsets coupled with syntactic filtering) (Dorr, 1997)—not syntactic alternations.

An example of an entry that we added to the database is the verb *oblige*. We have assigned a semantic representation and thematic grid to this verb, creating a new class 002—which we call *Coerce Verbs*—corresponding to verbs whose underlying meaning corresponds to "force to act". Because Levin's repository omits verbs taking clausal complements, several other verbs with a similar meaning fell into this class (e.g., *coerce*, *compel*, *persuade*) including some that were already included in the original system, but not in this class (e.g., *ask*). Thus, the LCS

mapping included X more verb-pattern pairs (such as Y), which we were not able to produce without manual inspection of the relevant verbs.

Database contains 50% more verbs and twice as many verb entries since the original framework of Levin. The result is that we can now parse constructions such as *She compelled him to eat* and *She asked him to eat*, which would not have been analyzable had we compiled our parsing lexicon on the basis of Levin's classes alone.

Levin's original proposal also does not contain semantic representations or thematic grids. When we built the LCS database, we examined each verb class carefully by hand to determine the underlying components of meaning unifying the members of that class. For example, the LCS representation that we generated for verbs in the *put* class includes components of meaning corresponding to "spatial placement in some manner," thus covering *dangle*, *hang*, *suspend*, etc.

From these hand-generated LCS representations, we derived our thematic grids—the same ones that are mapped onto our syntactic patterns. For example, position 1 (the highest leftmost argument in the LCS) is always mapped into the agent role of the thematic grid. The grids are organized into a thematic hierarchy that provides the basis for determining argument assignments, thus enhancing the generation process in ways that could not have been done previously with Levin's classes alone—e.g., producing constructions like *John sent a book to Paul* instead of constructions like *The book sent John to Paul*. Although the value of the thematic hierarchy seems most relevant to generation, the overall semantic/thematic hierarchical organization enables the automatic construction of lexicons that are equally suitable for both parsing and generation, thus reducing our overall lexical acquisition effort for both processes.

Beyond the above considerations, the granularity of the original Levin framework also was not adequate for our interlingual MT and lexical acquisition efforts. Our augmented form of this repository has brought about a more refined classification in which we are able to accommodate aspectual distinctions. We encode knowledge about aspectual features (e.g., *telicity*) in our LCS representations, thus sub-dividing the classes into more specific sub-classes. The tests used for this sub-division are purely *semantic* in nature, not syntactic. An example is the Dowty-style test "*He was X-ing* entails *He has X-ed*" (Dowty, 1979), where X is atelic (as in *run*) only if this entailment is considered valid by a human—and telic otherwise (as in *win*).

The inclusion of this type of knowledge allows us to refine Levin's classification significantly. An example is Class 35.6—*Ferret Verbs*: In Levin's original framework, this class conflated verbs occurring in different aspectual categories. Using the semantic tests above, we found that, in fact, these verbs should be divided as follows (Olsen et al., 1997):

Ferret Verbs: nose ferret tease (telic); seek (atelic)

The implication of this division for parsing is that the verbal arguments are constrained in a way that was not available to us in the original Levin-style classification—thus easing the job of the parser in choosing attachment points:

Telic:

∗He ferreted the truth from him.

He ferreted the truth out of him

Atelic:

He sought the truth from him.

∗He sought the truth out of him

Finally, Levin makes no claims as to the applicability of the English classes to other languages. Orienting our LCS database more toward semantic (aspectual) features rather than syntactic alternations has brought us closer to an interlingual representation that has now been demonstrably ported (quickly) to multiple languages including Arabic, Chinese, and Spanish. For example, telicity has been shown to be a crucial deciding feature in translating between divergence languages (Olsen et al., 1998), as in the translation of English *run across* as Spanish *cruzar corriendo*.

To summarize, our work is intended to: (1) Investigate the realization of a parsing lexicon from an LCS database that has developed from extensive semantic enhancements to an existing framework of verb classes and (2) Automate this technique so that it is directly applicable to LCS databases in other languages.

7. Future Work and Conclusions

Our ongoing work involves the following:

1. Using a subset of LDOCE-based lexicon by taking only the most frequent verb-pattern pairs in a big corpus: We expect that this approach will produce more realistic recall values.

2. Creating parsing lexicons for different languages: Once we have an automated mapping from the semantic lexicon to the set of syntactic patterns, we can use this method to create parsing lexicons from semantic lexicons that we already have available in other languages (Chinese, Spanish and Arabic).

3. Integration of these parsing lexicons in ongoing machine translation work (Habash and Dorr, 2001): We will feed the created lexicons into a parser and examine how successful the lexicons are. The same lexicons will also be used in our current clustering project.

Some of the ideas mentioned above are explored in detail in (Ayan and Dorr, 2002).

We conclude that it is possible to produce a parsing lexicon by projecting from LCS-based lexical entries—achieving precision and recall on a par with a syntactic lexicon (Minipar) encoded by hand specifically for English. The consequence of this result is that, as semantic lexicons become increasingly available for multiple languages (ours are now available in English, Chinese, and Arabic), we are able to produce parsing lexicons automatically for each language.

Acknowledgments

This work has been supported, in part, by ONR MURI Contract FCPO.810548265 and Mitre Contract 010418-7712.

8. References

Necip Fazil Ayan and Bonnie J. Dorr. 2002. Creating Parsing Lexicons From Semantic Lexicons Automatically and Its Applications. Technical report, University of Maryland, College Park, MD. Technical Report: LAMP-TR-084, CS-TR-4352, UMIACS-TR-2002-32.

Michael Brent. 1993. From Grammar to Lexicon: Unsupervised Learning of Lexical Syntax. *Computational Linguistics*, 19(2):243–262.

Ted Briscoe and John Carroll. 1997. Automatic extraction of subcategorization from corpora. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP-97)*, Washington, DC.

J. Carroll and C. Grover. 1989. The Derivation of a Large Computational Lexicon for English from LDOCE. In B. Boguraev and Ted Briscoe, editors, *Computational lexicography for natural language processing*, pages 117–134. Longman, London.

Bonnie J. Dorr. 1993. *Machine Translation: A View from the Lexicon*. The MIT Press, Cambridge, MA.

Bonnie J. Dorr. 1997. Large-Scale Dictionary Construction for Foreign Language Tutoring and Interlingual Machine Translation. *Machine Translation*, 12(4):271–322.

Bonnie J. Dorr. 2001. LCS Verb Database. Technical Report Online Software Database, University of Maryland, College Park, MD. http://www.umiacs.umd.edu/~bonnie/-LCS_Database_Documentation.html.

David Dowty. 1979. *Word Meaning in Montague Grammar*. Reidel, Dordrecht.

Dania Egedi and Patrick Martin. 1994. A Freely Available Syntactic Lexicon for English. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, Nara, Japan.

Ralph Grishman, Catherine Macleod, and Adam Meyers. 1994. Comlex Syntax: Building a Computational Lexicon. In *Proceedings of the COLING*, Kyoto.

Nizar Habash and Bonnie Dorr. 2001. Large-Scale Language Independent Generation Using Thematic Hierarchies. In *Proceedings of MT Summit VIII, Santiago de Compostella, Spain*.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago, IL.

Dekang Lin. 1993. Principle-Based Parsing without Overgeneration. In *Proceedings of ACL-93*, pages 112–120, Columbus, Ohio.

Dekang Lin. 1998. Dependency-Based Evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation*, Granada, Spain, May.

Christopher D. Manning. 1993. Automatic Acquisition of a Large Subcategorization Dictionary from Corpora. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 235–242, Columbus, Ohio.

R. Mitten. 1992. *Computer-Usable Version of Oxford Advanced Learner's Dictionary of Current English*. Oxford Text Archive.

Mari Broman Olsen, Bonnie J. Dorr, and Scott C. Thomas. 1997. Toward Compact Monotonically Compositional Interlingua Using Lexical Aspect. In *Proceedings of the Workshop on Interlinguas in MT, MT Summit, New Mexico State University Technical Report MCCS-97-314*, pages 33–44, San Diego, CA, October. Also available as UMIACS-TR-97-86, LAMP-TR-012, CS-TR-3858, University of Maryland.

Mari Broman Olsen, Bonnie J. Dorr, and Scott C. Thomas. 1998. Enhancing Automatic Acquisition of Thematic Structure in a Large-Scale Lexicon for Mandarin Chinese. In *Proceedings of the Third Conference of the Association for Machine Translation in the Americas, AMTA-98, in Lecture Notes in Artificial Intelligence, 1529*, pages 41–50, Langhorne, PA, October 28–31.

P. Procter. 1978. *Longman Dictionary of Contemporary English*. Longman, London.

Suzanne Stevenson and Paola Merlo. 2002a. A Multilingual Paradigm for Automatic Verb Classification. In *Proceedings of Association of Computational Linguistics*, Philadelphia, PA.

Suzanne Stevenson and Paola Merlo. 2002b. Automatic verb classification using distributions of grammatical features. In *Proceedings of the 9th Conference of the European Chapter of ACL*, pages 45–52, Bergen, Norway.