Adapting Existing Grammars: The XLE Experience

Ronald M. Kaplan and **Tracy Holloway King** and **John T. Maxwell III**
Palo Alto Research Center
Palo Alto, CA 94304 USA
{kaplan, thking, maxwell}@parc.com

Abstract

We report on the XLE parser and grammar development platform (Maxwell and Kaplan, 1993) and describe how a basic Lexical Functional Grammar for English has been adapted to two different corpora (newspaper text and copier repair tips).

1 Introduction

Large-scale grammar development platforms should be able to be used to develop grammars for a wide variety of purposes. In this paper, we report on the the XLE system (Maxwell and Kaplan, 1993), a parser and grammar development platform for Lexical Functional Grammars. We describe some of the strategies and notational devices that enable the basic English grammar developed for the ParGram project (Butt et al., 1999; Butt et al., 2002) to be adapted to two corpora with different properties.

1.1 The Corpora

The STANDARD Pargram English grammar covers the core phenomena of English (e.g., main and subordinate clauses, noun phrases, adjectives and adverbs, prepositional phrases, coordination; see (Butt et al., 1999)). We have built two different specialized grammars on top of this: the EUREKA grammar and the WSJ grammar.

The EUREKA grammar parses the Eureka corpus of copier repair tips, a collection of documents offering suggestions for how to diagnose and fix particular copier malfunctions. These informal and unedited documents were contributed by copier repair technicians, and the corpus is characterized by a significant amount of ungrammatical input (e.g., typos, incorrect punctuation, telegraphic sentences) and much technical terminology (1). The goal of parsing this corpus is to provide input to a semantics and world-knowledge reasoning application (Everett et al., 2001).

(1) a. (SOLUTION 27032 70) If exhibiting 10-132 faults replace the pre-fuser transport sensor (Q10-130).

b. (SOLUTION 27240 80) 4. Enter into the machine log, the changes that have been made.

The WSJ grammar covers the UPenn Wall Street Journal (WSJ) treebank sentences (Marcus et al., 1994). This corpus is characterized by long sentences with many direct quotes and proper names, (2a). In addition, for evaluation and training purposes we also parsed a version of this corpus marked up with labeled brackets and part-of-speech tags, as in (2b). Riezler et al. (2002) report on our WSJ parsing experiments.

(2) a. But since 1981, Kirk Horse Insurance Inc. of Lexington, Ky. has grabbed a 20% stake of the market.

b. But since 1981, [NP-SBJ Kirk Horse Insurance Inc. of Lexington, Ky.] has/VBZ_ grabbed/VBN_ [NP a 20% stake of the market].

2 Priority-based Grammar Specialization

The XLE system is designed so that the grammar writer can build specialized grammars by both extending and restricting another grammar (in our case the base grammar is the STANDARD Pargram English grammar). An LFG grammar is presented to the XLE system in a priority-ordered sequence of files containing phrase-structure rules, lexical entries, abbreviatory macros and templates, feature declarations, and finite-state transducers for tokenization and morphological analysis. XLE is applied to a single root file holding a CONFIGURATION that identifies all the other files containing relevant linguistic specifications, that indicates how

those components are to be assembled into a complete grammar, and that specifies certain parameters that control how that grammar is to be interpreted. A key idea is that there can be only one definition of an item of a given type with a particular name (e.g., there can be only one NP rule although that single rule can have many alternative expansions), and items in a higher priority file override lower priority items of the same type with the same name. This set up is similar to the priority-override scheme of the earlier LFG Grammar Writer's Workbench (Kaplan and Maxwell, 1996).

This arrangement makes it relatively easy to construct a specialized grammar from a pre-existing standard. The specialized grammar is defined by a CONFIGURATION in its own root file that specifies the relevant STANDARD grammar files as well as the new files for the specialized grammar. The files for the specialized grammar can also contain items of different types (phrase-structure rules, lexical entries, templates, etc.), and they are ordered with higher priority than the STANDARD files.

Consider the configuration for the EUREKA grammar. It specifies all of the STANDARD grammar files as well as its own rule, template, lexicon, and morphology files. A part of this configuration is shown in (3) (the *notationtemplates.lfg* are shared by all the languages' grammars, not just English).

(3)
```
FILES   ../standard/english-lexicons.lfg
        ../standard/english-rules.lfg
        ../standard/english-templates.lfg
        ../../common/notationtemplates.lfg
        english-eureka-morphconfig
        eureka-lexicons.lfg
        eureka-rules.lfg
        eureka-templates.lfg
```

This configuration specifies that the EUREKA rules, templates, and lexical entries are given priority over the STANDARD items by putting the special EUREKA files at the end of the list. Thus, if the *../standard/english-rules.lfg* and *eureka-rules.lfg* files both contain a rule expanding the NP category, the one from the STANDARD file will be discarded in favor of the EUREKA rule.

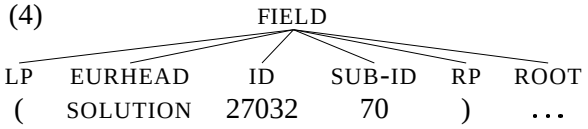
In the following subsections, we provide several illustrations of how simple overriding has been used for the EUREKA and WSJ grammar extensions.

2.1 Rules

The override convention makes it possible to: add rules (e.g., for new or idiosyncratic constructions); delete rules (e.g., to block constructions not found in the new corpus); and modify rules to allow different daughter sequences.

Rules may need to be added to allow for corpus-specific constructions. This is illustrated in the EUREKA corpus by the identifier information that precedes each sentence, as in (1). In order to parse this substring, a new category (FIELD) was defined with an expansion that covers the identifier information followed by the usual ROOT category of the STANDARD grammar. The top-level category is one of the parameters of a configuration, and the EUREKA CONFIGURATION specifies that FIELD instead of the STANDARD ROOT is the start-symbol of the grammar. Thus the EUREKA grammar produces the tree in (4) and functional-structure in (5) for (1a).

(4)



(5)

PRED	'replace<SUBJ, OBJ>'
SUBJ	[...]
OBJ	[...]
...	
FIELD	solution
TIP-ID	27032
SUB-TIP-ID	70

It is unusual in practice to need to delete a rule, i.e., to eliminate completely the possibility of expanding a given category of the STANDARD grammar. This is generally only motivated when the specialized grammar applies to a domain where certain constructions are rarely encountered, if at all. Although there has been no need to delete rules for the EUREKA and WSJ corpora, the override convention also provides a natural way of achieving this effect. For example, topicalization is extremely rare in the the Eureka corpus and the STANDARD topicalization rule sometimes introduces parsing inefficiency. This can be avoided by having the high priority EUREKA file replace the STANDARD rule with the one in (6).

(6) CPtop $\longrightarrow$.

This vacuous rule expands the CPtop category to the empty language, the language containing no strings;

so, this category is effectively removed from the grammar.

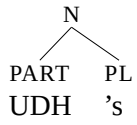
Perhaps the most common change is to make modifications to the behavior of existing rules. The most direct way of doing this is simply to define a new, higher priority expansion of the same left-hand category. Since XLE only allows a single rule for a given category, the old rule is discarded and the new one comes into play. The new rule can be arbitrarily different from the STANDARD one, but this is not typically the case. It is much more common that the specialized version incorporates most of the behavior of the original, with minor extensions or restrictions. One way of producing the modified behavior is to create a new rule that includes a copy of some or all of the STANDARD rule's right side along with new material, and to give the new definition higher priority than the old. For example, plurals in the Eureka corpus can be formed by the addition of *'s* instead of the usual *s*, as in (7).

(7) (CAUSE 27416 10) A 7mfd inverter motor capacitor was installed on an unknown number of UDH's.

In order to allow for this, the N rule was rewritten to allow a PL marker to optionally occur after any N, as in (8).

(8) N ⟶ copy of STANDARD N rule (PL)

As a result of this rule modification, *UDH's* in (7) will have the tree and functional-structure in (9).

(9) a.


b.
$$\begin{bmatrix} \text{PRED} & \text{'UDH'} \\ \text{NUM} & \text{pl} \end{bmatrix}$$

Copying material from one version to another is perhaps reasonable for relatively stable and simple rules, like the N rule, but this can cause maintainability problems with complicated rules in the STANDARD grammar that are updated frequently. An alternative strategy is to move the body of the STANDARD N rule to a different rule, e.g., Nbody, which in turn is called by the N rule in both the STANDARD and EUREKA grammars. The Nbody category can be supressed in the tree structure by invoking this rule as a macro (notationally indicated as @Nbody).

(10) N ⟶ @Nbody (PL).

Often the necessary modification can be made simply by redefining a macro that existing rules already invoke. Consider the ROOT rule, in (11).

(11) ROOT ⟶ { @DECL-BODY @DECL-PUNCT
| @INT-BODY @INT-PUNCT
| @HEADER }.

In the STANDARD grammar, the DECL-PUNCT macro is defined as in (12a). However, this must be modified in the EUREKA grammar because the punctuation is much sloppier and often does not occur at all; the EUREKA version is shown in (12b).

(12) a. DECL-PUNCT = { PERIOD
| EXCL-POINT }.

b. DECL-PUNCT = ({ PERIOD
| EXCL-POINT
| COLON
| SEMI-COLON }).

The modular specifications that macros and templates provide allow rule behavior to be modified without having to copy the parts of the rule that do not change.

XLE also has a mechanism for systematically modifying the behavior of all rules: the METARULEMACRO. For example, in order to parse labeled bracketed input, as in (2b), the WSJ grammar was altered so that constituents could optionally be surrounded by the appropriately labeled brackets. The METARULEMACRO is applied to each rule in the grammar and produces as output a modified version of that rule. This is used in the STANDARD grammar for coordination and to allow quote marks to surround any constituent. The METARULEMACRO is redefined for the WSJ to add the labeled bracketing possibilities for each rule, as shown in (13).

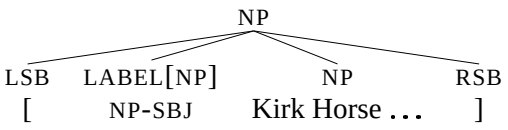
(13) METARULEMACRO(_CAT _BASECAT _RHS) =
{ LSB LABEL[_BASECAT] _CAT RSB
| copy of STANDARD coordination
| copy of STANDARD surrounding quote }.

The _CAT, _BASECAT, and _RHS are arguments to the METARULEMACRO that are instantiated to different values for each rule. _RHS is instantiated to the right-hand side of the rule, i.e., the rule expansion. _CAT and _BASECAT are two ways of representing the left-hand side of the rule. For simple categories the _CAT and _BASECAT are the same (e.g.

NP for the NP rule). XLE also allows for complex category symbols to specialize the expansion of particular categories in particular contexts. For example, the VP rule is parameterized for the form of its complement and its own form, so that VP[perf,fin] is one of the complex VP categories. When the METARULEMACRO applies to rules with complex left-side categories, _CAT refers to the category including the parameters and the _BASECAT refers to the category without the parameters. For the VP example, _CAT is VP[perf,fin] and _BASECAT is VP.

In the definition in (13), LSB and RSB parse the brackets themselves, while the LABEL[_BASECAT] parses the label in the bracketing and matches it to the label in the tree (NP in (2b)); the consituent itself is the _CAT. Thus, a label-bracketed NP is assigned the structure in (14).

(14)



These examples illustrate how the prioritized redefinition of rules and macros has enabled us to incorporate the STANDARD rules in grammars that are tuned to the special properties of the EUREKA and WSJ corpora.

2.2 Lexical Entries

Just as for rules, XLE's override conventions make it possible to: add new lexical items or new part-of-speech subentries for existing lexical items; delete lexical items; and modify lexical items. In addition to the basic priority overrides, XLE provides for "edit lexical entries" (Kaplan and Newman, 1997) that give finer control over the construction of the lexicon. Edit entries were introduced as a way of reconciling information from lexical databases of varying degrees of quality, but they are also helpful in tailoring a STANDARD lexicon to a specialized corpus. When working on specialized corpora, such as the Eureka corpus, modifications to the lexicon are extremely important for correctly handling technical terminology and eliminating word senses that are not appropriate for the domain.

Higher-priority edit lexical entries provide for operators that modify the definitions found in lower-priority entries. The operators can: add a subentry (+); delete a subentry (−); replace a subentry (!); or retain existing subentries (=). For example, the STANDARD grammar might have an entry for *button* as in (15).

(15) button !V @(V-SUBJ-OBJ %stem);
!N @(NOUN %stem);
ETC.

However, the EUREKA grammar might not need the V entry but might require a special partname N entry. Assuming that the EUREKA lexicons are given priority over the STANDARD lexicons, the entry in (16) would accomplish this.

(16) button −V ;
+N @(PARTNAME %stem);
ETC.

Note that the lexical entries in (15) and (16) end with ETC. This is also part of the edit lexical entry system. It indicates that other lower-priority definitions of that lexical item will be retained in addition to the new entries. For example, if in another EUREKA lexicon there was an adjective entry for *button* with ETC, the V, N, and A entries would all be used. The alternative to ETC is ONLY which indicates that only the new entry is to be used. In our *button* example, if an adjective entry was added with ONLY, the V and N entries would be removed, assuming that the adjective entry occurred in the highest priority lexicon. This machinery provides a powerful tool for building specialized lexicons without having to alter the STANDARD lexicons.

The EUREKA corpus contains a large number of names of copier parts. Due to their particular syntax and to post-syntactic processing requirements, a special lexical entry is added for each part name. In addition, the regular noun parse of these entries is deleted because whenever they occur in the corpus they are part names. A sample lexical is shown in (17); the ‘ is the escape character for the space.

(17) separator‘ finger
!PART-NAME @(PART-NAME %stem);
−N;
ETC.

The first line in (17) states that *separator finger* can be a PART NAME and when it is, it calls a template PART-NAME that provides relevant information for the functional-structure. The second line removes the N entry, if any, as signalled by the − before the category name.

Because of the non-context free nature of Lexical Functional Grammar, it sometimes happens that extensions in one part of the grammar require a corresponding adjustment in other rules or lexical entries. Consider again the EUREKA *'s* plurals. The part-name UDH is singular when it appears without the *'s* and thus the morphological tag +Sg is appended to it. In the STANDARD grammar, the tag +Sg has a lexical entry as in (18a) which states that +Sg is of category NNUM and assigns *sg* to its NUM. However, if this is used in the EUREKA grammar, the *sg* NUM specification will clash with the *pl* NUM specification when UDH appears with *'s*, as seen in (7). Thus, a new entry for +Sg is needed which has *sg* as a default value, as in (18b). The first line of (18b) states that NUM must exist but does not specify a value, while the second line optionally supplies a *sg* value to NUM; when the *'s* is used, this option does not apply since the form already has a *pl* NUM value.

(18) a. +Sg NNUM (↑ NUM)=sg

b. +Sg NNUM (↑ NUM)
((↑ NUM)=sg)

3 Tokenizing and Morphological Analysis

Tokenization and morphological analysis in XLE are carried out by means of finite state transductions. The STANDARD tokenizing transducer encodes the punctuation conventions of normal English text, which is adequate for many applications. However, the Eureka and WSJ corpora include strings that must be tokenized in non-standard ways. The Eureka part identifiers have internal punctuation that would normally cause a string to be broken up (e.g. the hyphen in PL1-B7), and the WSJ corpus is marked up with labeled brackets and part-of-speech tags that must also receive special treatment. An example of the WSJ mark-up is seen in (19).

(19) [NP-SBJ Lloyd's, once a pillar of the world insurance market,] is/VBZ_ being/VBG_ shaken/VBN_ to its very foundation.

Part-of-speech tags appear in a distinctive format, beginning with a / and ending with a _, with the intervening material indicating the content of the tag (*VBZ* for finite 3rd singular verb, *VBG* for a progressive, *VBN* for a passive, etc.). The tokenizing transducer must recognize this pattern and split the tags off as separate tokens. The tag-tokens must be available to filter the output of the morphological analyzer so that only verbal forms are compatible with the tags in this example and the adjectival reading of *shaken* is therefore blocked.

XLE tokenizing transducers are compiled from specifications expressed in the sophisticated Xerox finite state calculus (Beesley and Karttunen, 2002). The Xerox calculus includes the composition, ignore, and substitution operator discussed by Kaplan and Kay (1994) and the priority-union operator of Kaplan and Newman (1997). The specialized tokenizers are constructed by using these operators to combine the STANDARD specification with expressions that extend or restrict the standard behavior. For example, the ignore operator is applied to allow the part-of-speech information to be passed through to the morphology without interrupting the standard patterns of English punctuation.

XLE also allows separately compiled transducers to be combined at run-time by the operations of priority-union, composition, and union. Priority-union was used to supplement the standard morphology with specialized "guessing" transducers that apply only to tokens that would otherwise be unrecognized. Thus, a finite-state guesser was added to identify Eureka fault numbers (09-425), adjustment numbers (12-23), part numbers (606K2100), part list numbers (PL1-B7), repair numbers (2.4), tag numbers (P-102), and diagnostic code numbers (dC131). Composition was used to apply the part-of-speech filtering transducer to the output of the morphological analyzer, and union provided an easy way of adding new, corpus-specific terminology.

4 Optimality Marks

XLE supports a version of Optimality Theory (OT) (Prince and Smolensky, 1993) which is used to rank an analysis relative to other possible analyses (Frank et al., 2001). In general, this is used within a specific grammar to prefer or disprefer a construction. However, it can also be used in grammar extensions to delete or include rules or parts of rules.

The XLE implementation of OT works as follows.[1] OT marks are placed in the grammar and are associated with particular rules, parts of rules, or lexical entries. These marks are then ranked in the grammar CONFIGURATION. In addition to a simple ranking of constraints which states that a construction with a given OT mark is (dis)prefered to one

[1]The actual XLE OT implementation is more complicated than this, allowing for UNGRAMMATICAL and STOPPOINT marks as well. Only OT marks that are associated with NOGOOD are of interest here. For a full description, see (Frank et al., 2001).

without it, XLE allows the marks to be specified as NOGOOD. A rule or rule disjunct which has a NOGOOD OT mark associated with it will be ignored by XLE. This can be used for grammar extensions in that it allows a standard grammar to anticipate the variations required by special corpora without using them in normal circumstances.

Consider the example of the EUREKA 's plurals discussed in section 2.1. Instead of rewriting the N rule in the EUREKA grammar, it would be possible to modify it in the STANDARD grammar and include an OT mark, as in (20).

(20) N $\longrightarrow$ original STANDARD N rules
(PL: @(OT-MARK EUR-PLURAL)).

The CONFIGURATION files of the STANDARD and EUREKA grammars would differ in that the STANDARD grammar would rank the EUR-PLURAL OT mark as NOGOOD, as in (21a), while the EUREKA grammar would simply not rank the mark, as in (21b).

(21) a. STANDARD optimality order:
EUR-PLURAL NOGOOD ...

b. EUREKA optimality order:
NOGOOD ...

Given the OT marks, it would be possible to have one large grammar that is specialized by different OT rankings to produce the STANDARD, EUREKA, and WSJ variants. However, from a grammar writing perspective this is not a desirable solution because it becomes difficult to keep track of which constructions belong to standard English and are shared among all the specializations and which are corpus-specific. In addition, it does not distinguish a core set of slowly changing linguistic specifications for the basic patterns of the language, and thus does not provide a stable foundation that the writers of more specialized grammars can rely on.

5 Maintenance with Grammar Extensions

Maintenance is a serious issue for any large-scale grammar development activity, and the maintenance problems are compounded when multiple versions are being created perhaps by several different grammar writers. Our STANDARD grammar is now quite mature and covers all the linguistically significant constructions and most other constructions that we have encountered in previous corpus analysis. However, every now and then, a new corpus, even a specialized one, will evidence a standard construction that has not previously been accounted for. If specialized grammars were written by copying all the STANDARD files and then modifying them, the implementation of new standard constructions would tend to appear only in the specialized grammar. Our techniques for minimizing the amount of copying encourages us to implement new constructions in the STANDARD grammar and this makes them available to all other specializations.

If a new version of a rule for a specialized grammar is created by copying the corresponding STANDARD rule, changes later made to the special rule will not automatically be reflected in the STANDARD grammar, and vice versa. This is the desired behavior when adding unusual, corpus-specific constructions. However, if the non-corpus specific parts of the new rule are modified, these modifications will not migrate to the STANDARD grammar. To avoid this problem, the smallest rule possible should be modified in the specialized grammar, e.g., modifying the N head rule instead of the entire NP. For this reason, having highly modularized rules and using macros and templates helps in grammar maintenance both within a grammar and across specialized grammar extensions.

As seen above, the XLE grammar development platform provides a number of mechanisms to allow for grammar extensions without altering the core (STANDARD) grammar. However, there are still areas that could use improvement. For example, as mentioned in section 2, the CONFIGURATION file states which other files the grammar includes and how they are prioritized. The CONFIGURATION contains other information such as declarations of the governable grammatical functions, the distributive features, etc. As this information rarely changes with grammar extensions, it would be helpful for an extension configuration to incorporate by reference such additional parameters of the STANDARD configuration. Currently these declarations must be copied into each CONFIGURATION.

6 Discussion and Conclusion

As a result of the strategies and notational devices outlined above, our specialized grammars share substantial portions of the pre-existing STANDARD grammar. The statistics in table (22) give an indication of the size of the STANDARD grammar and of

the additional material required for the EUREKA and WSJ specializations. As can be seen from this table, the specialized grammars require a relatively small number of rules compared to the rules in the STANDARD grammar. The number of lines that the rules and lexical entries take up also provides a measure of the relative size of the specifications. The WSJ lexicons include many titles and proper nouns that may ultimately be moved to the STANDARD files. The table also shows the number of files called by the CONFIGURATION, as another indication of the size of the specifications. This number is somewhat arbitrary as separate files can be combined into a single multi-sectioned file, although this is likely to reduce maintainability and readability.

(22)

	STANDARD	EUREKA	WSJ
rules	310	32	14
lines:			
rules	6,539	425	894
lexicons	44,879	5,565	15,135
files	14	5	8

The grammars compile into a collection of finite-state machines with the number of states and arcs listed in table (23). The WSJ grammar compiles into the largest data structures, mainly because of its ability to parse labeled bracketed strings and part-of-speech tags, (2b). This size increase is the result of adding one disjunct in the METARULEMACRO and hence reflects only a minor grammar change.

(23)

	STANDARD	EUREKA	WSJ
states	4,935	5,132	8,759
arcs	13,268	13,639	19,695

In sum, the grammar specialization system used in XLE has been quite sucessful in developing corpus specific grammars using the STANDARD English grammar as a basis. A significant benefit comes from being able to distinguish truly unusual constructions that exist only in the specialized grammar from those that are (or should be) in the STANDARD grammar. This allows idiosyncratic information to remain in a specialized grammar while all the specialized grammars benefit from and contribute to the continuing development of the STANDARD grammar.

References

K. Beesley and L. Karttunen. 2002. *Finite-State Morphology: Xerox Tools and Techniques.* Cambridge University Press. To Appear.

M. Butt, T.H. King, M.-E. Niño, and F. Segond. 1999. *A Grammar Writer's Cookbook.* CSLI Publications, Stanford, CA.

M. Butt, H. Dyvik, T.H. King, H. Masuichi, and C. Rohrer. 2002. The parallel grammar project. In *Proceedings of COLING 2002.* Workshop on Grammar Engineering and Evaluation.

J. Everett, D. Bobrow, R. Stolle, R. Crouch, V. de Paiva, C. Condoravdi, M. van den Berg, and L. Polanyi. 2001. Making ontologies work for resolving redundancies across documents. *Communications of the ACM*, 45:55–60.

A. Frank, T. H. King, J. Kuhn, and J. T. Maxwell III. 2001. Optimality theory style constraint ranking in large-scale LFG grammars. In Peter Sells, editor, *Formal and Empirical Issues in Optimality Theoretic Syntax.* CSLI Publications, Stanford, CA.

R. Kaplan and M. Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20:331–378.

R. Kaplan and J. Maxwell. 1996. LFG Grammar Writer's Workbench. System documentation manual; available on-line at PARC.

R. Kaplan and P. Newman. 1997. Lexical resource conciliation in the Xerox Linguistic Environment. In *Proceedings of the ACL Workshop on Computational Environments for Grammar Development and Engineering.*

M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn treebank: Annotative predicate argument structure. In *ARPA Human Language Technology Workshop.*

J. Maxwell and R. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Lingusitics*, 19:571–589.

A. Prince and P. Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. RuCCS Technical Report #2, Rutgers University.

S. Riezler, T.H. King, R. Kaplan, D. Crouch, J. T. Maxwell, III, and M. Johnson. 2002. Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, University of Pennsylvania.*