Robust, Applied Morphological Generation

Guido Minnen **John Carroll** **Darren Pearce**
Cognitive and Computing Sciences
University of Sussex
Brighton BN1 9QH, UK
{*firstname.lastname*}*@cogs.susx.ac.uk*

Abstract

In practical natural language generation systems it is often advantageous to have a separate component that deals purely with morphological processing. We present such a component: a fast and robust morphological generator for English based on finite-state techniques that generates a word form given a specification of the lemma, part-of-speech, and the type of inflection required. We describe how this morphological generator is used in a prototype system for automatic simplification of English newspaper text, and discuss practical morphological and orthographic issues we have encountered in generation of unrestricted text within this application.

1 Introduction

Most approaches to natural language generation (NLG) ignore morphological variation during word choice, postponing the computation of the actual word forms to be output to a final stage, sometimes termed 'linearisation'. The advantage of this setup is that the syntactic/lexical realisation component does not have to consider all possible word forms corresponding to each lemma (Shieber et al., 1990). In practice, it is advantageous to have morphological generation as a postprocessing component that is separate from the rest of the NLG system. A benefit is that since there are no competing claims on the representation framework from other types of linguistic and non-linguistic knowledge, the developer of the morphological generator is free to express morphological information in a perspicuous and elegant manner. A further benefit is that localising morphological knowledge in a single component facilitates more systematic and reliable updating. From a software engineering perspective, modularisation is likely to reduce system development costs and increase system reliability. As an individual module, the morphological generator will be more easily shareable between several different NLG applications, and integrated into new ones. Finally, such a generator can be used on its own in other types of applications that do not contain a standard NLG syntactic/lexical realisation component, such as text simplification (see Section 3).

In this paper we describe a fast and robust generator for the inflectional morphology of English that generates a word form given a specification of a lemma, a part-of-speech (PoS) label, and an inflectional type. The morphological generator was built using data from several large corpora and machine readable dictionaries. It does not contain an explicit lexicon or word-list, but instead comprises a set of morphological generalisations together with a list of exceptions for specific (irregular) word forms. This organisation into generalisations and exceptions can save time and effort in system development since the addition of new vocabulary that has regular morphology does not require any changes to the generator. In addition, the generalisation-exception architecture can be used to specify—and also override—preferences in cases where a lemma has more than one possible surface word form given a particular inflectional type and PoS label.

The generator is packaged up as a Unix 'filter', making it easy to integrate into applications. It is based on efficient finite-state techniques, and is implemented using the widely available Unix Flex utility (a reimplementation of the AT&T Unix Lex tool) (Levine et al., 1992). The generator is freely available to the NLG research community (see Section 5 below).

The paper is structured as follows. Section 2 describes the morphological generator and eval-

uates its accuracy. Section 3 outlines how the generator is put to use in a prototype system for automatic simplification of text, and discusses a number of practical morphological and orthographic issues that we have encountered. Section 4 relates our work to that of others, and we conclude (Section 5) with directions for future work.

2 Morphological Generation

2.1 The Generator

The morphological generator covers the productive English affixes *s* for the plural form of nouns and the third person singular present tense of verbs, and *ed* for the past tense, *en* for the past participle, and *ing* for the present participle forms of verbs.[1] The generator is implemented in Flex.

The standard use of Flex is to construct 'scanners', programs that recognise lexical patterns in text (Levine et al., 1992). A Flex description—the high-level description of a scanner that Flex takes as input—consists of a set of 'rules': pairs of regular expression patterns (which Flex compiles into deterministic finite-state automata (Aho et al., 1986)), and actions consisting of arbitrary C code. Flex creates as output a C program which at run-time scans a text looking for occurrences of the regular expressions. Whenever it finds one, it executes the corresponding C code. Flex is part of the Berkeley Unix distribution and as a result Flex programs are very portable. The standard version of Flex works with any ISO-8559 character set; Unicode support is also available.

The morphological generator expects to receive as input a sequence of tokens of the form *lemma+inflection_label*, where *lemma* specifies the lemma of the word form to be generated, *inflection* specifies the type of inflection (i.e. *s*, *ed*, *en* or *ing*), and *label* specifies the PoS of the word form. The PoS labels follow the same pattern as in the Lancaster CLAWS tag sets (Garside et al., 1987; Burnard, 1995), with noun tags starting with *N*, etc. The symbols *+* and *_* are delimiters.

An example of a morphological generator rule is given in (1).

(1)
```
{A}+"s+s_N"
    {return(np_word_form(1,"es"));}
```

The left-hand side of the rule is a regular expression. The braces signify exactly one occurrence of an element of the character set abbreviated by the symbol `A`; we assume here that `A` abbreviates the upper and lower case letters of the alphabet. The next symbol `+` specifies that there must be a sequence of one or more characters, each belonging to the character set abbreviated by `A`. Double quotes indicate literal character symbols. The right-hand side of the rule gives the C code to be executed when an input string matches the regular expression. When the Flex rule matches the input *address+s_N*, for example, the C function `np_word_form` (defined elsewhere in the generator) is called to determine the word form corresponding to the input: the function deletes the inflection type and PoS label specifications and the delimiters, removes the last character of the lemma, and finally attaches the characters *es*; the word form generated is thus *addresses*.

Of course not all plural noun inflections are correctly generated by the rule in (1) since there are many irregularities and subregularities. These are dealt with using additional, more specific, rules. The order in which these rules are applied to the input follows exactly the order in which the rules appear in the Flex description. This makes for a very simple and perspicuous way to express generalizations and exceptions. For instance, the rule in (2) generates the plural form of many English nouns that originate from Latin, such as *stimulus*.

(2)
```
{A}+"us+s_N"
    {return(np_word_form(2,"i"));}
```

With the input *stimulus+s_N*, the output is *stimuli* rather than the incorrect **stimuluses* that would follow from the application of the more general rule in (1). By ensuring that this rule precedes the rule in (1) in the description, nouns such as *stimulus* get the correct plural form inflection. Some other words in this class, though, do not have the Latinate plural form (e.g. **boni* as a plural form of *bonus*); in these cases the generator contains rules specifying the correct forms as exceptions.

[1] We do not currently cover comparative and superlative forms of adjectives or adverbs since their productivity is much less predictable.

2.2 Inflectional Preferences

The rules constituting the generator do not necessarily have to be mutually exclusive, so they can be used to capture the inflectional morphology of lemmata that have more than one possible inflected form given a specific PoS label and inflectional type. An example of this is the multiple inflections of the noun *cactus*, which has not only the Latinate plural form *cacti* but also the English plural form *cactuses*. In addition, inflections of some words differ according to dialect. For example, the past participle form of the verb *to bear* is *borne* in British English, whereas in American English the preferred word form is *born*.

In cases where there is more than one possible inflection for a particular input lemma, the order of the rules in the Flex description determines the inflectional preference. For example, with the noun *cactus*, the fact that the rule in (2) precedes the one in (1) causes the generator to output the word form *cacti* rather than *cactuses* even though both rules are applicable.[2] It is important to note, though, that the generator will always choose between multiple inflections: there is no way for it to output all possible word forms for a particular input.[3]

2.3 Consonant Doubling

An important issue concerning morphological generation that is closely related to that of inflectional preference is consonant doubling. This phenomenon, occurring mainly in British English, involves the doubling of a consonant at the end of a lemma when the lemma is inflected. For example, the past tense/participle inflection of the verb *to travel* is *travelled* in British English, where the final consonant of the lemma is doubled before the suffix is attached. In American English the past tense/participle inflection of the verb *to travel* is usually spelt *traveled*. Consonant doubling is triggered on the basis of both orthographic and phonological information: when a word ends in one vowel followed by one consonant and the last part of the word is stressed, in general the consonant is doubled (Procter, 1995). However there are exceptions to this, and in any case the input to the morphological generator does not contain information about stress.

Consider the Flex rule in (3), where the symbols `C` and `V` abbreviate the character sets consisting of (upper and lower case) consonants and vowels, respectively.

(3) `{A}*{C}{V}"t+ed_V"`
`{return(cb_wordform(0,"t","ed"));}`

Given the input *submit+ed_V* this rule correctly generates *submitted*. However, the verb *to exhibit* does not undergo consonant doubling so this rule will generate, incorrectly, the word form *exhibitted*.

In order to ensure that the correct inflection of a verb is generated, the morphological generator uses a list of (around 1,100) lemmata that allow consonant doubling, extracted automatically from the British National Corpus (BNC; Burnard, 1995). The list is checked before inflecting verbs. Given the fact that there are many more verbs that do not allow consonant doubling, listing the verbs that do is the most economic solution. An added benefit is that if a lemma *does* allow consonant doubling but is not included in the list then the word form generated will still be correct with respect to American English.

2.4 Deriving the Generator

The morphological generator comprises a set of of approximately 1,650 rules expressing morphological regularities, subregularities, and exceptions for specific words; also around 350 lines of C/Flex code for program initialisation and defining the functions called by the rule actions. The rule set is in fact obtained by automatically reversing a morphological *analyser*. This is a much enhanced version of the analyser originally developed for the GATE system (Cunningham et al., 1996). Minnen and Carroll (Under review) describe in detail how the reversal is performed. The generator executable occupies around 700Kb on disc.

The analyser—and therefore the generator—includes exception lists derived from WordNet (version 1.5; Miller et al., 1993). In addition, we have incorporated data acquired semi-

[2] Rule choice based on ordering in the description can in fact be overridden by arranging for the second or subsequent match to cover a larger part of the input so that the longest match heuristic applies (Levine et al., 1992). But note that the rules in (1) and (2) will always match the same input span.

[3] Flex does not allow the use of rules that have identical left-hand side regular expressions.

automatically from the following corpora and machine readable dictionaries: the LOB corpus (Garside et al., 1987), the Penn Treebank (Marcus et al., 1993), the SUSANNE corpus (Sampson, 1995), the Spoken English Corpus (Taylor and Knowles, 1988), the Oxford Psycholinguistic Database (Quinlan, 1992), and the "Computer-Usable" version of the Oxford Advanced Learner's Dictionary of Current English (OALDCE; Mitton, 1992).

2.5 Evaluation

Minnen and Carroll (Under review) report an evaluation of the accuracy of the morphological generator with respect to the CELEX lexical database (version 2.5; Baayen et al., 1993). This threw up a small number of errors which we have now fixed. We have rerun the CELEX-based evaluation: against the past tense, past and present participle, and third person singular present tense inflections of verbs, and all plural nouns. After excluding multi-word entries (phrasal verbs, etc.) we were left with 38,882 out of the original 160,595 word forms. For each of these word forms we fed the corresponding input (derived automatically from the lemmatisation and inflection specification provided by CELEX) to the generator.

We compared the generator output with the original CELEX word forms, producing a list of mistakes apparently made by the generator, which we then checked by hand. In a number of cases either the CELEX lemmatisation was wrong in that it disagreed with the relevant entry in the *Cambridge International Dictionary of English* (Procter, 1995), or the output of the generator was correct even though it was not identical to the word form given in CELEX. We did not count these cases as mistakes. We also found that CELEX is inconsistent with respect to consonant doubling. For example, it includes the word form *pettifogged*,[4] whereas it omits many consonant doubled words that are much more common (according to counts from the BNC). For example, the BNC contains around 850 occurrences of the word form *programming* tagged as a verb, but this form is not present in CELEX. The form *programing* does occur in CELEX, but does not in the BNC. We did not count these cases as mistakes either. Of the remaining 359 mistakes, 346 concerned word forms that do not occur at all in the 100M words of the BNC. We categorised these as irrelevant for practical applications and so discarded them. Thus the *type* accuracy of the morphological analyser with respect to the CELEX lexical database is 99.97%. The *token* accuracy is 99.98% with respect to the 14,825,661 relevant tokens in the BNC (i.e. a rate of two errors per ten thousand words).

We tested the processing speed of the generator on a Sun Ultra 10 workstation. In order to discount program startup times (which are anyway only of the order of 0.05 seconds) we used input files of 400K and 800K tokens and recorded the difference in timings; we took the averages of 10 runs. Despite its wide coverage the morphological generator is very fast: it generates at a rate of more than 80,000 words per second.[5]

3 The Generator in an Applied System

3.1 Text Simplification

The morphological generator forms part of a prototype system for automatic simplification of English newspaper text (Carroll et al., 1999). The goal is to help people with aphasia (a language impairment typically occurring as a result of a stroke or head injury) to better understand English newspaper text. The system comprises two main components: an analysis module which downloads the source newspaper texts from the web and computes syntactic analyses for the sentences in them, and a simplification module which operates on the output of the analyser to improve the comprehensibility of the text. Syntactic simplification (Canning and Tait, 1999) operates on the syntax trees produced in the analysis phase, for example converting sentences in the passive voice to active, and splitting long sentences at appropriate points. A subsequent lexical simplification stage (Devlin and Tait, 1998) replaces difficult or rare content words with simpler synonyms.

The analysis component contains a morphological analyser, and it is the base forms of

[4] A rare word, meaning to be overly concerned with small, unimportant details.

[5] It is likely that a modest increase in speed could be obtained by specifying optimisation levels in Flex and gcc that are higher than the defaults.

words that are passed through the system; this eases the task of the lexical simplification module. The final processing stage in the system is therefore morphological generation, using the generator described in the previous section.

3.2 Applied Morphological Generation

We are currently testing the components of the simplification system on a corpus of 1000 news stories downloaded from the *Sunderland Echo* (a local newspaper in North-East England). In our testing we have found that newly encountered vocabulary only rarely necessitates any modification to the generator (or rather the analyser) source; if the word has regular morphology then it is handled by the rules expressing generalisations. Also, a side-effect of the fact that the generator is derived from the analyser is that the two modules have exactly the same coverage and are guaranteed to stay in step with each other. This is important in the context of an applied system. The accuracy of the generator is quite sufficient for this application; our experience is that typographical mistakes in the original newspaper text are much more common than errors in morphological processing.

3.3 Orthographic Postprocessing

Some orthographic phenomena span more than one word. These cannot be dealt with in morphological generation since this works strictly a word at a time. We have therefore implemented a final orthographic postprocessing stage. Consider the sentence:[6]

(4) **Brian Cookman is the attraction at the* ***King 's Arms*** *on Saturday night and* ***he will*** *be back on Sunday night for* ***a acoustic*** *jam session.*

This is incorrect orthographically because the determiner in the final noun phrase should be *an*, as in *an acoustic jam session.* In fact *an* must be used if the following word starts with a vowel sound, and *a* otherwise. We achieve this, again using a filter implemented in Flex, with a set of general rules keying off the next word's first letter (having skipped any intervening sentence-internal punctuation), together with a list of exceptions (e.g. *heir*, *unanimous*) collected using the pronunciation information in the OALDCE, supplemented by further cases (e.g. *unidimensional*) found in the BNC. In the case of abbreviations or acronyms (recognised by the occurrence of non-word-initial capital letters and trailing full-stops) we key off the pronunciation of the first letter considered in isolation.

Similarly, the orthography of the genitive marker cannot be determined without taking context into account, since it depends on the identity of the last letter of the preceding word. In the sentence in (4) we need only eliminate the space before the genitive marking, obtaining *King's Arms.* But, following the newspaper style guide, if the preceding word ends in *s* or *z* we have to 'reduce' the marker as in, for example, *Stacey Edwards' skilful fingers.*

The generation of contractions presents more of a problem. For example, changing *he will* to *he'll* would make (4) more idiomatic. But there are cases where this type of contraction is not permissible. Since these cases seem to be dependent on syntactic context (see Section 4 below), and we have syntactic structure from the analysis phase, we are in a good position to make the correct choice. However, we have not yet tackled this issue and currently take the conservative approach of not contracting in any circumstances.

4 Related Work

We are following a well-established line of research into the use of finite-state techniques for lexical and shallow syntactic NLP tasks (e.g. Karttunen et al. (1996)). Lexical transducers have been used extensively for morphological analysis, and in theory a finite-state transducer implementing an analyser can be reversed to produce a generator. However, we are not aware of published research on finite-state morphological generators (1) establishing whether in practice they perform with similar efficiency to morphological analysers, (2) quantifying their type/token accuracy with respect to an independent, extensive 'gold standard', and (3) indicating how easily they can be integrated into larger systems. Furthermore, although a number of finite-state compilation toolkits (e.g. Karttunen (1994)) are publicly available or can

[6] This sentence is taken from the story "The demise of Sunderland's Vaux Breweries is giving local musicians a case of the blues" published in the *Sunderland Echo* on 26 August 1999.

be licensed for research use, associated large-scale linguistic descriptions—for example English morphological lexicons—are usually commercial products and are therefore not freely available to the NLG research community.

The work reported here is also related to work on lexicon representation and morphological processing using the DATR representation language (Cahill, 1993; Evans and Gazdar, 1996). However, we adopt less of a theoretical and more of an engineering perspective, focusing on morphological generation in the context of wide-coverage practical NLG applications. There are also parallels to research in the two-level morphology framework (Koskenniemi, 1983), although in contrast to our approach this framework has required exhaustive lexica and hand-crafted morphological (unification) grammars in addition to orthographic descriptions (van Noord, 1991; Ritchie et al., 1992). The SRI Core Language Engine (Alshawi, 1992) uses a set of declarative segmentation rules which are similar in content to our rules and are used in reverse to generate word forms. The system, however, is not freely available, again requires an exhaustive stem lexicon, and the rules are not compiled into an efficiently executable finite-state machine but are only interpreted.

The work that is perhaps the most similar in spirit to ours is that of the LADL group, in their compilation of large lexicons of inflected word forms into finite-state transducers (Mohri, 1996). The resulting analysers run at a comparable speed to our generator and the (compacted) executables are of similar size. However, a full form lexicon is unwieldy and inconvenient to update, and a system derived from it cannot cope gracefully with unknown words because it does not contain generalisations about regular or subregular morphological behaviour.

The morphological components of current widely-used NLG systems tend to consist of hard-wired procedural code that is tightly bound to the workings of the rest of the system. For instance, the Nigel grammar (Matthiessen, 1984) contains Lisp code that classifies verb, noun and adjective endings, and these classes are picked up by further code inside the KPML system (Bateman, 2000) itself which performs inflectional generation by stripping off variable length trailing strings and concatenating suffixes. All morphologically subregular forms must be entered explicitly in the lexicon, as well as irregular ones. The situation is similar in FUF/SURGE, morphological generation in the SURGE grammar (Elhadad and Robin, 1996) being performed by procedures which inspect lemma endings, strip off trailing strings when appropriate, and concatenate suffixes.

In current NLG systems, orthographic information is distributed throughout the lexicon and is applied via the grammar or by hard-wired code. This makes orthographic processing difficult to decouple from the rest of the system, compromising maintainability and ease of reuse. For example, in SURGE, markers for *a/an* usage can be added to lexical entries for nouns to indicate that their initial sound is consonant- or vowel-like, and is contrary to what their orthography would suggest. (This is only a partial solution since adjectives, adverbs—and more rarely other parts of speech—can follow the indefinite article and thus need the same treatment). The appropriate indefinite article is inserted by procedures associated with the grammar. In DRAFTER-2 (Power et al., 1998), an *a/an* feature can be associated with any lexical entry, and its value is propagated up to the NP level through leftmost rule daughters in the grammar (Power, personal communication). Both of these systems interleave orthographic processing with other processes in realisation. In addition, neither has a mechanism for stating exceptions for whole subclasses of words, for example those starting *us* followed by a vowel—such as *use* and *usual*—which must be preceded by *a*. KPML appears not to perform this type of processing at all.

We are not aware of any literature describing (practical) NLG systems that generate contractions. However, interesting linguistic research in this direction is reported by Pullum and Zwicky (In preparation). This work investigates the underlying syntactic structure of sentences that block auxiliary reductions, for example those with VP ellipsis as in (5).

(5) **She's usually home when he's.*

5 Conclusions

We have described a generator for English inflectional morphology. The main features of the generator are:

wide coverage and high accuracy It incorporates data from several large corpora and machine readable dictionaries. An evaluation has shown the error rate to be very low.

robustness The generator does not contain an explicit lexicon or word-list, but instead comprises a set of morphological generalisations together with a list of exceptions for specific (irregular) words. Unknown words are very often handled correctly by the generalisations.

maintainability and ease of use The organisation into generalisations and exceptions can save development time since addition of new vocabulary that has regular morphology does not require any changes to be made. The generator is packaged up as a Unix filter, making it easy to integrate into applications.

speed and portability The generator is based on efficient finite-state techniques, and implemented using the widely available Unix Flex utility.

freely available The morphological generator and the orthographic postprocessor are freely available to the NLG research community. See <http://www.cogs.susx.ac.uk/lab/nlp/carroll/morph.html>.

In future work we intend to investigate the use of phonological information in machine readable dictionaries for a more principled solution to the consonant doubling problem. We also plan to further increase the flexibility of the generator by including an option that allows the user to choose whether it has a preference for generating British or American English spelling.

Acknowledgements

This work was funded by UK EPSRC project GR/L53175 'PSET: Practical Simplification of English Text', and by an EPSRC Advanced Fellowship to the second author. The original version of the morphological analyser was kindly provided to us by the University of Sheffield GATE project. Chris Brew, Dale Gerdemann, Adam Kilgarriff and Ehud Reiter have suggested improvements to the analyser/generator. Thanks also to the anonymous reviewers for insightful comments.

References

Alfred Aho, Ravi Sethi, and Jeffrey Ullman. 1986. *Compilers: Principles, Techniques and Tools*. Addison-Wesley.

Hiyan Alshawi, editor. 1992. *The Core Language Engine*. MIT Press, Cambridge, MA.

Harald Baayen, Richard Piepenbrock, and Hedderik van Rijn. 1993. *The CELEX Lexical Database (CD-ROM)*. Linguistic Data Consortium, University of Pennsylvania, Philadelphia, PA, USA.

John Bateman. 2000. *KPML (Version 3.1) March 2000*. University of Bremen, Germany, <http://www.fb10.uni-bremen.de/anglistik/langpro/kpml/README.html>.

Lou Burnard. 1995. Users reference guide for the British National Corpus. Technical report, Oxford University Computing Services.

Lynne Cahill. 1993. Morphonology in the lexicon. In *Proceedings of the 6th Conference of the European Chapter of the Association for Computational Linguistics*, pages 87–96, Utrecht, The Netherlands.

Yvonne Canning and John Tait. 1999. Syntactic simplification of newspaper text for aphasic readers. In *Proceedings of the ACM SIGIR Workshop on Customised Information Delivery*, Berkeley, CA, USA.

John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying English text for language impaired readers. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Bergen, Norway.

Hamish Cunningham, Yorick Wilks, and Robert Gaizauskas. 1996. GATE—a General Architecture for Text Engineering. In *Proceedings of the 16th Conference on Computational Linguistics*, Copenhagen, Denmark.

Siobhan Devlin and John Tait. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. In (Nerbonne, 1998).

Michael Elhadad and Jacques Robin. 1996. An overview of SURGE: A reusable comprehensive syntactic realization component. Technical Report 96-03, Dept of Mathematics and Computer Science, Ben Gurion University, Israel.

Roger Evans and Gerald Gazdar. 1996. DATR: a language for lexical knowledge representation. *Computational Linguistics*, 22.

Roger Garside, Geoffrey Leech, and Geoffrey Sampson. 1987. *The computational analysis of English: a corpus-based approach*. Longman, London.

Lauri Karttunen, Jean-Pierre Chanod, Gregory Grefenstette, and Anne Schiller. 1996. Regular expressions for language engineering. *Natural Language Engineering*, 2(4):305–329.

Lauri Karttunen. 1994. Constructing lexical transducers. In *Proceedings of the 14th Conference on Computational Linguistics*, pages 406–411, Kyoto, Japan.

Kimmo Koskenniemi. 1983. Two-level model for morphological analysis. In *8th International Joint Conference on Artificial Intelligence*, pages 683–685, Karlsruhe, Germany.

John Levine, Tony Mason, and Doug Brown. 1992. *Lex & Yacc*. O'Reilly and Associates, second edition.

Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Christian Matthiessen. 1984. Systemic Grammar in computation: The Nigel case. In *Proceedings of the 1st Conference of the European Chapter of the Association for Computational Linguistics*, pages 155–164, Pisa, Italy.

George Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, Katherine Miller, and Randee Tengi. 1993. *Five Papers on WordNet*. Princeton University, Princeton, N.J.

Guido Minnen and John Carroll. Under review. *Fast and robust morphological processing tools for practical NLP applications*.

Roger Mitton. 1992. A description of a computer-usable dictionary file based on the Oxford Advanced Learner's Dictionary of Current English. Available at <ftp://ota.ox.ac.uk/pub/ota/public/dicts/710/text710.doc>.

Mehryar Mohri. 1996. On some applications of finite-state automata theory to natural language processing. *Natural Language Engineering*, 2(1):61–80.

John Nerbonne, editor. 1998. *Linguistic Databases*. Lecture Notes. CSLI Publications, Stanford, USA.

Richard Power, Donia Scott, and Roger Evans. 1998. What You See Is What You Meant: direct knowledge editing with natural language feedback. In *Proceedings of the 13th Biennial European Conference on Artificial Intelligence (ECAI 98)*, Brighton, UK.

Paul Procter. 1995. *Cambridge International Dictionary of English*. Cambridge University Press.

Geoffrey Pullum and Arnold Zwicky. In preparation. Licensing of prosodic features by syntactic rules: the key to auxiliary reduction. First version presented to the Annual Meeting of the Linguistic Society of America, Chicago, Illinois, January 1997. Available at <http://www.lsadc.org/web2/99modabform.htm>.

Philip Quinlan. 1992. *The Oxford Psycholinguistic Database*. Oxford University Press.

Graeme Ritchie, Graham Russell, Alan Black, and Stephen Pulman. 1992. *Computational morphology: practical mechanisms for the English lexicon*. MIT Press.

Geoffrey Sampson. 1995. *English for the computer*. Oxford University Press.

Stuart Shieber, Gertjan van Noord, Robert Moore, and Fernando Pereira. 1990. Semantic head-driven generation. *Computational Linguistics*, 16(1):7–17.

Lita Taylor and Gerry Knowles. 1988. Manual of information to accompany the SEC Corpus: the machine-readable corpus of spoken English. Manuscript, University of Lancaster, UK.

Gertjan van Noord. 1991. Morphology in MiMo2. Manuscript, University of Utrecht, The Netherlands.