172. Computational Linguistics

Contents

1 Computational Linguistics

It is useful to distinguish in Computational Linguistics between applications and modules. Applications are geared toward a specific user-oriented goal (e.g., automatic translation, or dialogue with an information system), whereas modules are necessary in a wide range of applications. For example, syntactic parsing as a module is useful in the development of both Machine Translation systems and Dialogue Systems. Modules can be characterised by a transformation between different linguistic representation levels, e.g. from text to speech, or from a string of words to a tree structure representing the sentence's structure. The goal of much work in Computational Linguistics is to design efficient and accurate computational models for such transformations. In developing a syntactic *parser*, for example, we may build a model using a grammar, a lexicon and a heuristic search procedure which together transform strings of words (the input representation) into labeled trees representing the syntactic structure (the output representation).

In this article, I will treat morphology as one such module, which can operate in two directions. In morphological analysis, a complex word form is transformed into a string of morphemes with a characterisation of the structural relations between the different morphemes (possibly represented in a tree structure), and its stem or citation form. In morphological synthesis (or generation), a stem or root form of a morphological paradigm and a set of grammatical features is taken as input, and the corresponding complex word form is generated.

Morphological Analysis	
surprisingly	(((surprise)Verb ing)Adjective ly)Adverb
took	(take)Verb-Past
Morphological Synthesis	
establish (past participle)	established
clean (superlative)	cleanest

At a general level, all morphological processing modules will need a lexical database associating morphemes with linguistic information, a model of the combination possibilities of morphemes and the effects thereof on spelling and pronunciation (mostly in the form of *rules*), and a search procedure using this information to actually generate or analyze words.

In this article, I will briefly discuss the kind of models that have been proposed for designing the morphological module in Computational Linguistics applications, go into the different application areas where morphological modules are used, and give an overview of tools which could help morphologists in their research. I will also provide pointers to some recent work on *learning of morphology* which seems to attract the attention of both linguistic and computational morphology. There are several good introductions to the field of Computational Linguistics in general (Jurafsky and Martin, 2000; Manning and Schütze, 1999; Allen, 1995).

2 Models of Morphology in Computational Linguistics

As Computational Morphology (CM) focuses on developing models that achieve one of the morphological mappings discussed earlier (segmenting a string into its parts and disambiguating the parts in morphological analysis, construction of a string on the basis of lexical and morphological information in morphological synthesis), theoretical linguistic distinctions like inflection versus derivation do not play an important role. Formalisms developed for modeling morphology focus on the concrete construction processes involved (concatenation, Ablaut, root-and-template interleaving, suppletion etc.), on the spelling and phonological changes these processes produce, and on the constraints under which all this occurs (morphotactics). Issues of productivity will be reflected in CM by a particular choice of the division of labour between lexical storage and rule-based processing. We will only briefly describe the main types of models here. More technical and broader overviews of the field of CM can be found elsewhere (Sproat, 1992; Sproat, 2000).

2.1 The Role of Lexical Databases

In any CM model, there is a trade-off between the contents of the lexical database and the size of the rule set. In languages like English with a relatively poor morphology, it is feasible to construct a lexical database with complete paradigms of complex word forms rather than morphemes (a full form lexicon). The construction can be done semi-autmatically (by morphological synthesis, which is easier than analysis), leading to an analysis by synthesis approach. All inflectional and most of the derivational processes can then be solved by lexical retrieval rather than computation. However, for most languages this is not a feasible solution. For example, compounding is extremely productive in languages like German and Dutch, and for languages like Turkish and Finnish, inflection and derivation cannot be solved by this "pre-computation" and lexical storage approach. Moreover, the approach fails for words (stems) not contained in the lexicon.

For implementation, lexical databases tend to be represented as *letter tries* (Fredkin, 1960). A trie is a tree datastructure with nodes representing a position in a word, and arcs leaving a node representing letters that can follow the corresponding position in the word. The root node represents the start position, and individual words are paths in the trie. Leaf nodes represent the lexical information associated with the word the path of which ends at that node. This way the redundancy in the spelling of words is used to compress the lexicon into a data structure that is small and that can be searched efficiently.

Figure 2.1 is an example of the trie structure representing the five words *ask asking away be became*. Note how the redundancy is removed.

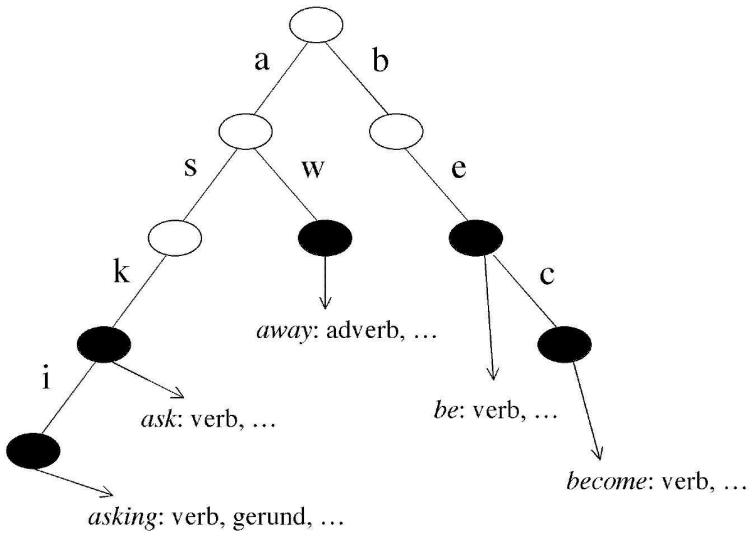


Figure 1: Example of a trie data structure containing the lexical items *ask asking away be became*

2.2 Finite-State Morphology

The most influential model developed for morphological analysis and synthesis to date is *finite-state morphology* (FSM), also called two-level morphology. It was developed by Koskenniemi (Koskenniemi, 1983; Koskenniemi, 1984) for Finnish and other languages, inspired by unpublished work by Martin Kay and Ron Kaplan, published only much later (Kaplan and Kay, 1994). The model consists of a trie lexicon structure and a set of rules implemented as a finite-state transducer, and can perform both analysis and synthesis.

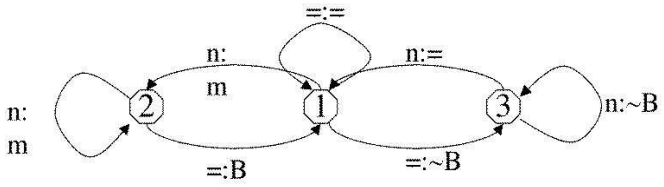
The most important ideas in this approach were that all or most morphological phenomena can be described with regular expressions, and that the morphological mappings can be described with only two levels: a lexical representation, and a surface (spelling or speech) representation. Traditional linguistic descriptions like generative phonology (Chomsky and Halle, 1968) made used of an ordered series of rewrite rules with intermediate representations to transform lexical representations into surface forms, where each rule works on the output of the previous rule, thereby giving rise to complex rule interactions. In addition, the expressive power of the formalism used to describe each rule (context-sensitive grammar) was obviously higher than needed, and in a computational perspective, this type of rule only works in one direction (from lexical to surface). A further advantage of finite-state approaches is that

the computational machinery involved is language-independent, and not *ad hoc* for a specific language.

In FSM, each rule is implemented as a bi-directional Finite-State Transducer linking the lexical and surface representation, making it useful for both analysis and synthesis, and all rules are applied in parallel. The base unit of a two-level rule or constraint is a pair of symbols, one from the alphabet of the lexical representation, one from the alphabet of the surface representation. E.g. the pair $n : m$ (a lexical n corresponding to a surface m), or $V : 0$ (a lexical vowel deleted at the surface level). See Figure 2.2(a), which represents an assimilation rule: a lexical n has to be realised as a surface m when followed by a bilabial consonant ($B = m, b, or p$, the symbol = can refer to any symbol). Figure 2.2(b) and 2.2(c) show the corresponding Finite-State Transducers as a network (transitions show how states are allowed to change) and as a transition table. The automatic compilation of two-level rules into FSTs is not trivial, but can be done. The strength of the approach is that two-level rules can refer to the lexical as well as to the surface representations for defining their context.

(a) n:m ⇔ _ =:B (where B = m, b, or p)

(b)



(c)

	n m	= B	n =	n ~B	= ~B	= =
1	2		3			1
2	2	1				
3				3	1	

Figure 2: Three representations of a FSM rule; two-level rule (a), transition network (b), and transition table (c).

An FSM model consists of a set of rules of this type. In morphological synthesis, the rules get a series of lexical symbols as input and produce all surface forms allowed by the rules, in morphological analysis, they get a surface form as input, and produce all possible lexical representations (constrained by the lexicon). The lexicon in FSM consists of a list of stem forms, and a number of affix lexicons (lists of inflectional and derivational suffixes). Lexicon entries contain associated lexical information, including a list of pointers to continuation lexica (lexica which can follow this lexicon entry).

The model has been applied to many languages and many morpho-phonological phenomena, and has proven to be an elegant and practical approach to CM. Many overviews, tutorial material, extensions, and references are available (Gazdar, 1985; Ritchie et al., 1992; Antworth, 1990). An extension for morphologically motivated phenomena like Umlaut is presented in Trost (1991). Despite its success, FSM is not without problems; the lexicon

system leads to redundant representations, and some processes like reduplication and root-and-template morphology cannot be described without complicating the formalism.

The success of finite-state approaches in CM no doubt contributed to the investigation of the applicability of finite-state methods to syntactic analysis and in applications such as shallow semantic interpretation (Abney, 1996; Karttunen et al., 1996; Roche and Schabes, 1997). Together with statistical approaches, finite-state methods dominate Computational Linguistics today.

2.3 Hierarchical Lexicons

Another popular approach to computational morphology makes use of the concept of taxonomies and inheritance of properties to represent morphological knowledge. The basic insight here is that irregular words normally deviate only in a few characteristics from regular words. E.g., an irregular verb like *run* is just like a regular verb like *work* (compare run, runs, running to work, works, working), except that the past tense and past participle are formed in a different way. This "elsewhere condition" type of reasoning, so common in linguistic morphology, can be modeled elegantly with inheritance networks (Daelemans, Smedt, and Gazdar, 1992). Although there are many general-purpose knowledge representation and programming languages (of the frame-based or object-oriented type) that allow the implementation of this type of reasoning, the most important exponent of this approach is DATR (Evans and Gazdar, 1996), a special-purpose programming language for lexical knowledge representation. DATR was guided in its design by formal adequacy goals (explicit declarative semantics and explicit theory of inference), and notational adequacy goals (it should be expressive enough to describe all relevant generalizations. Another concern was efficient implementation). DATR fragments have been developed for the morphology of various languages.

Both approaches are complementary as the finite-state approach is limited in the mechanisms it allows for handling morphotactics, which is what most DATR work focusses on. Not all models can be assigned to one of these approaches; one successful morphological analysis module in the context of speech synthesis, DECOMP (Allen, Hunnicutt, and Klatt, 1987), uses ad hoc weights to improve disambiguation. I will return to the problem of morphological ambiguity resolution in the section on morphology learning.

3 Morphology in Language Technology

Morphological analysis is a basic component in many language technology applications. In Word Processing applications, for example, the accuracy of hyphenation, spelling checking, and grammar checking is highly dependent on the presence of some form of morphological analysis. Morphological processing can also be considered a core component in any language processing system, from speech synthesis over information retrieval to machine translation.

Depending on the sophistication of the morphological processing involved, different terms have been used in language technology. The term *stemming* or *suffix stripping* is used to refer to a sort of "poor man's" morphological analysis involving simple rules (often without a lexicon) to reduce an inflectional form of a word to its stem. A classical example of this approach, adapted to many languages is the Porter stemmer (Porter, 1980). This approach is of course only feasible with some accuracy for languages like English which have a simple morphology. The term *lemmatization* refers to a more advanced process in which a complex wordform is reduced to its lemma (or citation form) and the possible morphosyntactic

classes it can have as retrieved from the lexicon and deduced from the rules. In this case the structure of the word is not further analyzed. So whereas full morphological analysis would analyze *optimizations* as $((((optimal)Adjective-ize)Verb-ation)Noun-s)Noun-plural$, a lemmatizer would output $optimalizationNoun-plural$, depending on the contents of the lexicon, and the definition of the rules, of course.

3.1 Word Processing

Automatic hyphenation is the process of splitting words at the end of lines in order to minimize whitespace when right justification is used. Although the process is conventional, it is mostly based on linguistic units like syllables and morphemes. In languages like English where hyphenation is morpheme-based, morphological segmentation is required. However, given the poor morphology of English, a good computational solution is to use a set of splitting patterns found in a dictionary rather than full morphological analysis (Liang, 1983). In a language like Dutch, however, where hyphenation is based on syllabification and morphological structure, the situation is much more complex. It is fairly easy to implement the basic syllabification rules for Dutch (based on the maximal onset principle and a language-specific rule avoiding syllables ending in a short vowel). It is sufficient to collect a list of possible syllable onsets. Interestingly, these rules are overridden by morphological rules. In compounds and some derivations, e.g. with the suffix $-achtig$ (transl. $-ly$), the morphological boundary overrides the syllable boundary, giving rise to oppositions like *groe-nig* (groen+ig, greeny) versus *groen-achtig* (groen+achtig, greeny) where in the first case the maximal onset principle is preserved, and in the second case the morphological boundary has precedence over the syllable boundary in hyphenation. A fairly accurate morphological analysis is required to solve this problem in principle, in this case splitting patterns will necessarily be error-prone because of the high incidence of new compounds in Dutch, as can be witnessed every day in Dutch newspapers (Daelemans, 1988).

Spelling Checking is based on a very simple principle to detect errors. Given a list of words of the language, every word encountered in a text which does not belong to the list is a spelling error. In languages with a productive morphology, this leads to an annoying *overkill*; the software continuously flags correct words as errors. Especially compounding is a problem. Extremely productive in languages like German and Dutch, this morphological process is responsible for the creation of many complex words which are used *ad hoc* in a text, and never gain enough frequency to warrant their inclusion in the word list. Morphological analysis is the only way to solve this problem. Similar arguments hold for languages with an extensive derivational or inflectional morphology like Finnish or Turkish, where storage of all word forms in a word list is impossible. Kukich (1992) describes the role of natural language processing in spelling correction.

In **Grammar Checkers**, software that checks the grammatical correctness of sentences in a text, morphological analysis is essential to be able to detect agreement errors, e.g. between subject and verb in many languages, and between modifiers and nouns in languages like German and French.

Processes very similar to morphological analysis are also necessary in word processing of languages like Chinese, and Japanese and Korean when written with Chinese characters, where word boundaries are not marked by spaces or other typographical means. A further good example is Vietnamese: it is written in Latin characters, but white space indicates syllable boundaries rather than word boundaries.

3.2 Module in Larger Systems

Morphological analysis is a necessary component in complete Natural Language Processing systems (for speech recognition and synthesis, language understanding, language generation, language translation, information retrieval etc.), mostly as a means to increase the *lexical coverage* of such a system.

For example, a syntactic parser needs lexical information about every word in a sentence to be parsed. In case words are not in the lexicon, morphological analysis helps extract useful lexical information from these unknown word forms (often complex forms of known words), increasing the "virtual" coverage of the lexicon. In speech processing, morphological analysis is important as well. In Speech Recognition as a means to keep the recognizer lexicon small, in Speech Synthesis as a means to increase lexical coverage and solve ambiguous proncunciations like *th* in *nothing* versus *anthill*.

With the availability of the WWW, **Information Retrieval** has become a ubiquitous technology. In search engines, keywords can be input to retrieve a number of documents containing them, ranked according to relevance according to statistical or heuristic measures. The reliability of an information retrieval engine is measured in terms of *recall* (how many of the documents relevant for my query did I get) and *precision* (how many of the documents returned by the system actually were relevant). Morphological processing is one way of increasing the recall of search engines. By expanding a keyword to all its morphologically related forms (by morphological synthesis) on the basis of its stem (found by morphological analysis), a wider range of (possibly relevant) documents is found, increasing recall.

4 Morphology Learning

The last decade, statistical approaches have started dominating the field of computational linguistics (Manning and Schütze, 1999). The field has evolved from *deductive* to *inductive*. Recently, machine learning methods have been added to the tools of inductive computational linguistics. Machine Learning is a subfield of Artificial Intelligence concerned with the design of algorithms that learn from examples (Mitchell, 1997). When used for building a linguistic model explaining some set of data, machine learning algorithms and linguists share the same task and purpose, which makes the approach potentially interesting for linguistics.

Machine Learning algorithms can be *supervised*, in which case they get examples of available input and required output, or *unsupervised*, in which case they only get examples of available input, and have to figure out useful groupings or clusters of the data. The goal of a learning approach is to use the examples to find useful generalizations about the input-output mapping to be learned. In an example of supervised machine learning applied to morphological analysis (den Bosch and Daelemans, 1999), a morphological analysis system is induced based on a large set of examples of complex words and their corresponding morphological analysis from the CELEX lexical database (Baayen, Piepenbrock, and van Rijn, 1993). It can reconstruct the analyses it used for training, and apply the same systematicity to previously unseen complex words with high accuracy. The approach has been applied to English, German, and Dutch, and shows that even complex spelling changes can be handled in this classification-based way. Daelemans, Berck, and Gillis (1997) is an example of a similar approach for morphological synthesis. In this case, the output of the learning is a rule system for diminutive formation in Dutch which is very similar to linguistic solutions proposed for the problem.

Supervised learning methods solve a problem which approaches like FST leave basically unsolved: the disambiguation problem. Even in languages with a minimally complex morphology, a morphological analysis system can lead to many possible analyses (different segmentations into morphemes, different assignments of grammatical classes to morphemes), many of which are spurious. A supervised machine learning approach implicitly uses frequency information from the data it was trained on to make probabilistic disambiguation decisions.

Especially in *unsupervised* learning of morphology, a lot of progress has been achieved the last few years. Starting from a list of complex words, several unsupervised learning techniques have been experimented with to automatically extract information like lists of stems and affixes, and of how they can be combined. (Goldsmith, 2001; Kazakov and Manandhar, 2001; Yarowsky and Wicentowski, 2000). An exciting new approach in between supervised and unsupervised learning would start from a bilingual corpus, semi-automatically aligned. Suppose a morphological analyzer exists for German and not for Dutch. Once a bilingual aligned corpus German - Dutch is collected, the analyses generated for German can be projected to the Dutch part of the corpus and used to induce a morphological analyzer for Dutch (Yarowsky and Ngai, 2001).

5 Morphology Tools for Linguists

Some of the approaches used in CM are at the heart of what constitutes a linguistic approach to morphology (finding the right generalizations and morpheme inventories for describing the morphology of a language, and looking for formalism which allow describing them in sufficient detail). It is therefore not surprising that many CM researchers have tried to build reusable tools that will be of interest to linguists as well. In this section a (probably incomplete) overview of the most important ones is given.

Finite-state morphology is a well-developed field which has spun off many useful tools. The Xerox Research Center in Europe (XRCE) has developed tools for morphological analysis in many languages, based on finite-state technology and is also active in research in this area. Evan Antworth of SIL provides a useful two-level morphology software package called PC-KIMMO. Finite-state tools for inflectional morphological analysis and synthesis of English implemented using widely-available unix utilities are available from the University of Sussex (Minnen, Carroll, and Pearce, 2001).

Also many implementations for DATR have been developed. Dafydd Gibbon provides an implementation called Zdatr, and the developers of DATR (Roger Evans and Gerald Gazdar) maintain a web portal with tutorial information, references, and references to implementations.

Simpler application-oriented tools like Porter stemmers are widely available as well.

Many implementations of Machine Learning algoritms are available as well, but are generally not directly usable for linguistic research. Systematic evaluation is only just starting (Maxwell, 2002) as well. BOAS is a user-friendly environment for the development of morphological anayzers which makes use of Machine Learning (Oflazer, Nirenburg, and McShan, 2001). Interactive demonstrations of the supervised learning approach of (den Bosch and Daelemans, 1999) are available as well via the demonstrations section of their website.

6 References

Abney, Steven. 1996. Partial parsing via finite-state cascades. In John Carroll and Ted Briscoe, editors, *Proceedings of the ESSLLI '96 Robust Parsing Workshop*, pages 8–15, Prague. ESSLLI.

Allen, J., M. S. Hunnicutt, and D. Klatt. 1987. *From Text to Speech: The MITalk System.* Cambridge University Press, Cambridge, England.

Allen, James. 1995. *Natural Language Understanding — 2nd Edition.* The Benjamin/Cummings Publishing Company, Redwood City, CA.

Antworth, Evan L. 1990. *PC-KIMMO: a two-level processor for morphological analysis.* Number 16 in Occasional Publications in Academic Computing. Summer Institute of Linguistics, Dallas, TX.

Baayen, R. H., P. Piepenbrock, and H. van Rijn, editors. 1993. *The CELEX Lexical Database (CD-ROM).* Linguistic Data Consortium, University of Pennsylvania, Philadelphia (PA).

Chomsky, Noam and Morris Halle. 1968. *The Sound Pattern of English.* Harper & Row, New York, NY.

Daelemans, W., P. Berck, and S. Gillis. 1997. Data mining as a method for linguistic analysis: Dutch diminutives. *Folia Linguistica*, XXXI(1–2):57–75.

Daelemans, Walter, Koenraad De Smedt, and Gerald Gazdar. 1992. Inheritance in natural language processing. *Computational Linguistics*, 18(2):205–218.

Daelemans, Walter M. P. 1988. Automatic hyphenation: linguistics versus engineering. In F. J. Heyvaert & Frieda Steurs, editor, *Worlds behind Words.* Leuven University Press, Leuven, pages 347–364.

den Bosch, A. Van and W. Daelemans. 1999. Memory-based morphological analysis. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 285–292, San Francisco, CA. Morgan Kaufmann.

Evans, Roger and Gerald Gazdar. 1996. DATR: A language for lexical knowledge representation. *Computational Linguistics*, 22(2):167–216.

Fredkin, E. 1960. Trie memory. *Communications of the ACM*, pages 490–499.

Gazdar, Gerald. 1985. Finite state morphology. *Linguistics*, 23:597–607.

Goldsmith, John. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

Jurafsky, Daniel and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall, Englewood Cliffs, New Jersey.

Kaplan, Ron and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378.

Karttunen, L., J. Chanod, G. Grefenstette, and A. Schiller. 1996. Regular expressions for language engineering. *Natural Language Engineering*, 2(4):305–338.

Kazakov, Dimitar and Suresh Manandhar. 2001. Unsupervised learning of word segmentation rules with genetic algorithms and inductive logic programming. *Machine Learning*, 43:121–162.

Koskenniemi, Kimmo. 1983. Two-level model for morphological analysis. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, Los Alamos, CA. Morgan Kaufmann.

Koskenniemi, Kimmo. 1984. A general computational model for word-form recognition and production. In *Proceedings 10th International Conference on Computational Linguistics*, Stanford, CA. ICCL.

Kukich, Karen. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439.

Liang, Franklin Mark. 1983. *Word Hy-phen-a-tion by Com-put-er.* Ph.D. thesis, Stanford University, Stanford, CA.

Manning, Christopher D. and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, Massachusetts.

Maxwell, Mike. 2002. Resources for morphology learning and evaluation. In Manuel González Rodríguez and Carmen Paz Suárez Araujo, editors, *LREC 2002: Third International Conference on Language Resources and Evaluation*, volume III, pages 967–974, Paris. ELRA.

Minnen, Guido, John Carroll, and David Pearce. 2001. Applied morphological processing of english. *Natural Language Engineering*, 7(3):207–223.

Mitchell, Tom M. 1997. *Machine Learning.* McGraw-Hill, New York, NY.

Oflazer, Kemal, Sergei Nirenburg, and Marjorie McShan. 2001. Bootstrapping morphological analyzers by combining human elicitation and machine learning. *Computational Linguistics*, 27(1).

Porter, M. F. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Ritchie, Graeme D., Graham J. Russell, Alan W. Black, and Stephen G. Pulman. 1992. *Computational Morphology: practical mechanisms for the English lexicon.* MIT Press, Cambridge, MA.

Roche, Emmanuel and Yves Schabes, editors. 1997. *Finite-State Language Processing.* MIT Press, Cambridge, MA.

Sproat, Richard. 2000. Lexical analysis. In Robert Dale, Hermann Moisl, and Harold Somers, editors, *Handbook of Natural Language Processing.* Marcel Dekker, New York and Basel, pages 37–57.

Sproat, Richard W. 1992. *Morphology and Computation.* MIT Press, Cambridge, MA.

Trost, Harald. 1991. X2morph: A morphological component based on augmented two-level morphology. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 1024–1030, San Francisco, CA. Morgan Kaufmann.

Yarowsky, D. and G. Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proceedings of NAACL-2001*, pages 200–207, San Francisco, CA. Morgan Kaufmann.

Yarowsky, D. and R. Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of ACL-2000*, pages 207–216, San Francisco, CA. Morgan Kaufmann.

Walter Daelemans, Antwerpen (Belgium)