LATTICE-BASED WORD IDENTIFICATION IN CLARE

David M. Carter
SRI International
Cambridge Computer Science Research Centre
23 Millers Yard
Cambridge CB2 1RQ, U.K.

dmc@cam.sri.com

ABSTRACT

I argue that because of spelling and typing errors and other properties of typed text, the identification of words and word boundaries in general requires syntactic and semantic knowledge. A lattice representation is therefore appropriate for lexical analysis. I show how the use of such a representation in the CLARE system allows different kinds of hypothesis about word identity to be integrated in a uniform framework. I then describe a quantitative evaluation of CLARE's performance on a set of sentences into which typographic errors have been introduced. The results show that syntax and semantics can be applied as powerful sources of constraint on the possible corrections for misspelled words.

1 INTRODUCTION

In many language processing systems, uncertainty in the boundaries of linguistic units, at various levels, means that data are represented not as a well-defined sequence of units but as a lattice of possibilities. It is common for speech recognizers to maintain a lattice of overlapping word hypotheses from which one or more plausible complete paths are subsequently selected. Syntactic parsing, of either spoken or written language, frequently makes use of a chart or well-formed substring table because the correct bracketing of a sentence cannot (easily) be calculated deterministically. And lattices are also often used in the task of converting Japanese text typed in kana (syllabic symbols) to kanji; the lack of interword spacing in written Japanese and the complex morphology of the language mean that lexical items and their boundaries cannot be reliably identified without applying syntactic and semantic knowledge (Abe *et al*, 1986).

In contrast, however, it is often assumed that, for languages written with interword spaces, it is sufficient to group an input character stream deterministically into a sequence of words, punctuation symbols and perhaps other items, and to hand this sequence to the parser, possibly after word-by-word morphological analysis. Such an approach is sometimes adopted even when typographically complex inputs are handled; see, for example, Futrelle *et al*, 1991.

In this paper I observe that, for typed input, spaces do not necessarily correspond to boundaries between lexical items, both for linguistic reasons and because of the possibility of typographic errors. This means that a lattice representation, not a simple sequence, should be used throughout front end (preparsing) analysis. The CLARE system under development at SRI Cambridge uses such a representation, allowing it to deal straightforwardly with combinations or multiple occurrences of phenomena that would be difficult or impossible to process correctly under a sequence representation. As evidence for the performance of the approach taken, I describe

an evaluation of CLARE's ability to deal with typing and spelling errors. Such errors are especially common in interactive use, for which CLARE is designed, and the correction of as many of them as possible can make an appreciable difference to the usability of a system.

The word identity and word boundary ambiguities encountered in the interpretation of errorful input often require the application of syntactic and semantic knowledge on a phrasal or even sentential scale. Such knowledge may be applied as soon as the problem is encountered; however, this brings major problems with it, such as the need for adequate lookahead, and the difficulties of engineering large systems where the processing levels are tightly coupled. To avoid these difficulties, CLARE adopts a staged architecture, in which indeterminacy is preserved until the knowledge needed to resolve it is ready to be applied. An appropriate representation is of course the key to doing this efficiently.

2 SPACES AND WORD BOUNDARIES

In general, typing errors are not just a matter of one intended input token being miskeyed as another one. Spaces between tokens may be deleted (so that two or more intended words appear as one) or inserted (so that one word appears as two or more). Multiple errors, involving both spaces and other characters, may be combined in the same intended or actual token. A reliable spelling corrector must allow for all these possibilities, which must, in addition, be distinguished from the use of correctly-typed words that happen to fall outside the system's lexicon.

However, even in the absence of "noise" of this kind, spaces do not always correspond to lexical item boundaries, at least if lexical items are defined in a way that is most convenient for grammatical purposes. For example, "special" forms such as telephone numbers or e-mail addresses, which are common in many domains, may contain spaces. In CLARE, these are analysed using regular expressions (cf Grosz *et al*, 1987), which may include space characters. When such an expression is realised, an analysis of it, connecting non-adjacent vertices if it contains spaces, is added to the lattice.

The complexities of punctuation are another source of uncertainty: many punctuation symbols have several uses, not all of which necessarily lead to the same way of segmenting the input. For example, periods may indicate either the end of a sentence or an abbreviation, and slashes may be simple word-internal characters (e.g. *X11/NeWS*) or function lexically as disjunctions, as in

> *[1] I'm looking for suggestions for vendors to deal with/avoid.* [1]

Here, the character string "*with/avoid*", although it contains no spaces, represents three lexical items that do not even form a syntactic constituent.

CLARE's architecture and formalism allow for all these possibilities, and, as an extension, also permit multiple-token phrases, such as idioms, to be defined as equivalent to other tokens or token sequences. This facility is especially useful when CLARE is being tailored for use in a particular domain, since it allows people not expert in linguistics or the CLARE grammar to extend grammatical coverage in simple and approximate, but often practically important, ways. For example, if an application developer finds that inputs such as "What number of employees have cars?" are common, but that the construction "what number of ..." is not handled by the grammar, he can define the sequence "what number of" as equivalent to "how many". This will provide an extension of coverage without the developer needing to know how any of the phrases involved are treated in the grammar. Extending the grammar is, of course, a more thorough solution if the expertise is available; the phrasal equivalence suggested here will not, for example,

[1] These two examples are taken from the Sun-spots computer bulletin board.

cope correctly with the query "What number of the employees have cars?".

3 CLARE'S PROCESSING STAGES

The CLARE system is intended to provide language processing capabilities (both analysis and generation) and some reasoning facilities for a range of possible applications. English sentences are mapped, via a number of stages, into logical representations of their literal meanings, from which reasoning can proceed. Stages are linked by well-defined representations. The key intermediate representation is that of *quasi logical form* (QLF; Alshawi, 1990, 1992), a version of slightly extended first order logic augmented with constructs for phenomena such as anaphora and quantification that can only be resolved by reference to context. The unification of declarative linguistic data is the basic processing operation.

The specific task considered in this paper is the process of mapping single sentences from character strings to QLF. Two kinds of issue are therefore not discussed here. These are the problem of segmenting a text into sentences and dealing with any markup instructions (cf Futrelle *et al*, 1991), which is logically prior to producing character strings; and possible context-dependence of the lexical phenomena discussed, which would need to be dealt with after the creation of QLFs.

In the analysis direction, CLARE's front end processing stages are as follows.

1. A sentence is divided into a sequence of *clusters* separated by white space.
2. Each cluster is divided into one or more *tokens*: words (possibly inflected), punctuation characters, and other items. Tokenization is nondeterministic, and so a lattice is used at this and subsequent stages.
3. Each token is analysed as a sequence of one or more *segments*. For normal lexical items, these segments are morphemes. The lexicon proper is first accessed at this stage.
4. A variety of strategies for *error recovery* (including but not limited to spelling/typing correction) are attempted on tokens for which no segmentation could be found. Edges without segmentations are then deleted; if no complete path remains, sentence processing is abandoned.
5. Further edges, possibly spanning non-adjacent vertices, are added to the lattice by the phrasal equivalence mechanism discussed earlier.

Morphological, syntactic and semantic stages then apply to produce one or more QLFs. These are checked for adherence to sortal (selectional) restrictions, and, possibly with the help of user intervention, one is selected for further processing.

Because tokenization is nondeterministic and does not involve lexical access, it will produce many possible tokens that cannot be further analysed. If sentence [1] above were processed, *with/avoid* would be one such token. It is important that analyses are found for as many tokens and token sequences as possible, but that error recovery, especially if it involves user interaction, is not attempted unless really necessary. More generally, the system must decide which techniques to apply to which problem tokens, and how the results of doing so should be combined.

CLARE's token segmentation phase therefore attempts to find analyses for all the single tokens in the lattice, and for any special forms, which may include spaces and therefore span multiple tokens. Next, a series of *recovery methods*, which may be augmented or reordered by the application developer, are applied. *Global* methods apply to the lattice as a whole, and are intended to modify its contents or create required lexicon entries on a scale larger than the individual token. *Local* methods apply only to single still-unanalysed tokens, and may either supply analyses for them

or alter them to other tokens. The default methods, all of which may be switched on or off using system commands, supply facilities for inferring entries through access to an external machine-readable dictionary; for defining sequences of capitalized tokens as proper names; for spelling correction (described in detail in the next section); and for interacting with the user who may suggest a replacement word or phrase or enter the VEX lexical acquisition subsystem (Carter, 1989) to create the required entries.

After a method has been applied, the lattice is, if possible, pruned: edges labelled by unanalysed tokens are provisionally removed, as are other edges and vertices that then do not lie on a complete path. If pruning succeeds (i.e. if at least one problem-free path remains) then token analysis is deemed to have succeeded, and unanalysed tokens (such as *with/avoid*) are forgotten; any remaining global methods are invoked, because they may provide analyses for token *sequences*, but remaining local ones are not. If full pruning does not succeed, any subpath in the lattice containing more unrecognized tokens than an alternative subpath is eliminated. Subpaths containing tokens with with non-alphabetic characters are penalized more heavily; this ensures that if the cluster "`boooks,`" is input, the token sequence "`boooks ,`" (in which "`boooks`" is an unrecognized token and "`,`" is a comma) is preferred to the single token "`boooks,`" (where the comma is part of the putative lexical item). The next method is then applied.[2]

4 SEGMENTATION AND SPELLING CORRECTION

A fairly simple affix-stripping approach to token segmentation is adopted in CLARE because inflectional morphological changes in English tend not to be complex enough to warrant more powerful, and potentially less efficient, treatments such as two-level morphology (Koskenniemi, 1983). Derivational morphological relationships typically involve semantic peculiarities as well, necessitating the definition of derived words in the lexicon in their own right.

The rules for dividing clusters into tokens have the same form as those for segmenting tokens into morphemes, and are processed by the same mechanism. Thus "`,`", like, say, "`ed`", is defined as a suffix, but one that is treated by the grammar as a separate word rather than a bound morpheme. Rules for punctuation characters are very simple because no spelling changes are ever involved. However, the possessive ending "`'s`" is treated as a separate word in the CLARE grammar to allow the correct analysis of phrases such as "the man in the corner's wife", and spelling changes can be involved here. Like segmentation, tokenization can yield multiple results, mainly because there is no reason for a complex cluster like `Mr.` or `King's` not also to be defined as a lexical item.

One major advantage of the simplicity of the affix-stripping mechanism is that spelling correction can be interleaved directly with it. Root forms in the lexicon are represented in a discrimination net for efficient access (cf Emirkanian and Bouchard, 1988). When the spelling corrector is called to suggest possible corrections for a word, the number of simple errors (of deletion, insertion, substitution and transposition; e.g. Pollock and Zamora, 1984) to assume is given. Normal segmentation is just the special case of this with the number of errors set to zero. The mechanism nondeterministically removes affixes from each end of the word, postulating errors if appropriate, and then looks up the resulting string in the discrimination net, again considering the possibility of error.[3]

[2] In fact, for completeness, CLARE allows the application of two or more methods in tandem and will combines the results without any intermediate pruning. This option would be useful if, in a given application, two sources of knowledge were deemed to be about equally reliable in their predictions.

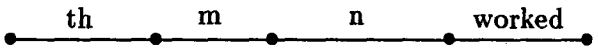
[3] This is the reverse of Veronis' (1988) algorithm, where roots are matched before affixes. However, it

Interleaving correction with segmentation like this promotes efficiency in the following way. As in most other correctors, only up to two simple errors are considered along a given search path. Therefore, either the affix-stripping phase or the lookup phase is fairly quick and produces a fairly small number of results, and so the two do not combine to slow processing down. Another beneficial consequence of the interleaving is that no special treatment is required for the otherwise awkward case where errors overlap morpheme boundaries; thus *desigend* is corrected to *designed* as easily as *deisgned* or *designde* are.

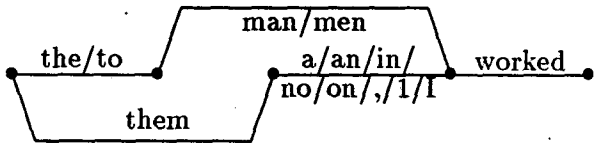
If one or more possible corrections to a token are found, they may either be presented to the user for selection or approval, or, if the number of them does not exceed a preset threshold, all be preserved as alternatives for disambiguation at the later syntactic or semantic stages. The lattice representation allows multiple-word corrections to be preserved along with single-word ones.

It is generally recognized that spelling errors in typed input are of two kinds: competence errors, where the user does not know, or has forgotten, how to spell a word; and performance errors, where the wrong sequence of keys is hit. CLARE's correction mechanism is oriented towards the latter. Other work (e.g. Veronis, 1988, Emirkanian and Bouchard, 1988, van Berkel and De Smedt, 1988) emphasizes the former, often on the grounds that competence errors are both harder for the user to correct and tend to make a worse impression on a human reader. However, Emirkanian and Bouchard identify the many-to-one nature of French spelling-sound correspondence as responsible for the predominance of such errors in that language, which they say does not hold in English; and material typed to CLARE tends to be processed further (for database access, translation, etc) rather than reproduced for potentially embarrassing human consumption. A performance-error approach also has the practical advantage of not depending on extensive linguistic knowledge; and many competence errors can be detected by a performance approach, especially if some straightforward adjustments (e.g. to prefer doubling to other kinds of letter insertion) are made to the algorithm.

As well as coping quite easily with morpheme boundaries, CLARE's algorithm can also handle the insertion or deletion of word boundary spaces. For the token *witha*, CLARE postulates both *with* and *with a* as corrections, and (depending on the current switch settings) both may go into the lattice. The choice will only finally be made when a QLF is selected on sortal and other grounds after parsing and semantic analysis. For the token pair *nev er*, CLARE postulates the single correction *never*, because this involves assuming only one simple error (the insertion of a space) rather than two or more to "correct" each token individually. Multiple overlapping possibilities can also be handled; the input *Th m n worked* causes CLARE to transform the initial lattice



into a corrected lattice containing analyses of the words shown here:



The edges labelled "them" and "man/men" are constructed first by the "global" spelling correction method, which looks for possible corrections across token boundaries. The edge for the token "m" is then removed because, given that it connects only to errorful tokens on both sides, it cannot form part of any potentially optimal path through the lattice. Corrections are, however, sought for "th" and

seems easier and more efficient to match affixes first, because then the hypothesized root can be looked up without having to allow for any spelling changes; and if both prefixes and suffixes are to be handled, as they are in CLARE, there is no obvious single starting point for searching for the root first.

"n" as single tokens when the local spelling correction method is invoked. The corrected lattice then undergoes syntactic and semantic processing, and QLFs for the sequences "the man worked" and "the men worked", but not for any sequence starting with "them" or "to", are produced.

5 AN EVALUATION

To assess the usefulness of syntactico-semantic constraints in CLARE's spelling correction, the following experiment, intended to simulate performance (typographic) errors, was carried out. Five hundred sentences, of up to ten words in length, falling within CLARE's current core lexical (1600 root forms) and grammatical coverage were taken at random from the LOB corpus. These sentences were passed, character by character, through a channel which transmitted a character without alteration with probability 0.99, and with probability 0.01 introduced a simple error. The relative probabilities of the four different kinds of error were deduced from Table X of Pollock and Zamora, 1984; where a new character had to be inserted or substituted, it was selected at random from the original sentence set. This process produced a total of 102 sentences that differed from their originals. The average length was 6.46 words, and there were 123 corrupted tokens in all, some containing more than one simple error. Because longer sentences were more likely to be changed, the average length of a changed sentence was some 15% more than that of an original one.

The corrupted sentence set was then processed by CLARE with only the spelling correction recovery method in force and with no user intervention. Up to two simple errors were considered per token. No domain-specific or context-dependent knowledge was used.

Of the 123 corrupted tokens, ten were corrupted into other known words, and so no correction was attempted. Parsing failed in nine of these cases; in the tenth, the corrupted word made as much sense as the original out of discourse context. In three further cases, the original token was not suggested as a correction; one was a special form, and for the other two, alternative corrections involved fewer simple errors. The corrections for two other tokens were not used because a corruption into a known word elsewhere in the same sentence caused parsing to fail.

Only one correction (the right one) was suggested for 59 of the remaining 108 tokens. Multiple-token correction, involving the manipulation of space characters, took place in 24 of these cases.

This left 49 tokens for which more than one correction was suggested, requiring syntactic and semantic processing for further disambiguation. The average number of corrections suggested for these 49 was 4.57. However, only an average of 1.69 candidates (including, because of the way the corpus was selected, all the right ones) appeared in QLFs satisfying selectional restrictions; thus only 19% of the wrong candidates found their way into any QLF. If, in the absence of frequency information, we take all candidates as equally likely, then syntactic and semantic processing reduced the average entropy from 1.92 to 0.54, removing 72% of the uncertainty (see Carter, 1987, for a discussion of why entropy is the best measure to use in contexts like this).

When many QLFs are produced for a sentence, CLARE orders them according to a set of scoring functions encoding syntactic and semantic preferences. For the 49 multiple-candidate tokens, removing all but the best-scoring QLF(s) eliminated 7 (21%) of the 34 wrong candidates surviving to the QLF stage; however, it also eliminated 5 (10%) of the right candidates. It is expected that future development of the scoring functions will further improve these figures, which are summarized in Table 1.

The times taken to parse lattices containing multiple spelling candidates reflect the characteristics of CLARE's parser, which uses a

Stage	Right cand's	Wrong cand's	Average number
Suggested	49	175	4.57
In any QLF	49	34	1.69
In best-scoring QLF(s)	44	27	1.45

Table 1: Correction candidates for the 49 multiple-candidate tokens

backtracking, left-corner algorithm and stores well-formed constituents so as to avoid repeating work where possible. In general, when a problem token appears late in the sentence and/or when several candidate corrections are syntactically plausible, the lattice approach is several times faster than processing the alternative strings separately (which tends to be very time-consuming). When the problem token occurs early and has only one plausible correction, the two methods are about the same speed.

For example, in one case, a corrupted token with 13 candidate corrections occurred in sixth position in an eight-word sentence. Parsing the resulting lattice was three times faster than parsing each alternative full string separately. The lattice representation avoided repetition of work on the first six words. However, in another case, where the corrupted token occurred second in an eight-word sentence, and had six candidates, only one of which was syntactically plausible, the lattice representation was no faster, as the incorrect candidates in five of the strings led to the parse being abandoned early.

An analogous experiment was carried out with 500 sentences from the same corpus which CLARE could *not* parse. 131 of the sentences, with average length 7.39 words, suffered the introduction of errors. Of these, only seven (5%) received a parse. Four of the seven received no sortally valid QLFs, leaving only three (2%) "false positives". This low figure is consistent with the results from the originally parseable sentence set; nine out of the ten corruptions into known words in that experiment led to parse failure, and only 19% of wrong suggested candidates led to a sortally valid QLF. If, as those figures suggest, the replacement of one word by another only rarely maps one sentence inside coverage to another, then a corresponding replacement on a sentence *outside* coverage should yield something within coverage even more rarely, and this does appear to be the case.

6 CONCLUSIONS

These experimental results suggest that general syntactic and semantic information is an effective source of constraint for correcting typing errors, and that a conceptually fairly simple staged architecture, where word identity and word boundary ambiguities are only resolved when the relevant knowledge is ready to be applied, can be acceptably efficient. The lattice representation also allows the system to deal cleanly with word boundary uncertainty not caused by noise in the input.

A fairly small vocabulary was used in the experiment. However, these words were originally selected on the basis of frequency of occurrence, so that expanding the lexicon would involve introducing proportionately fewer short words than longer ones. Mistyped short words tend to be the ones with many correction candidates, so the complexity of the problem should grow less fast than might be expected with vocabulary size. Furthermore, more use could be made of statistical information: relative frequency of occurrence could be used as a criterion for pruning relatively unlikely correction candidates, as could more sophisticated statistics in the suggestion algorithm, along the lines of Kernighan *et al* (1990). Phonological knowledge, to allow competence errors to be tackled more directly, would provide another useful source of constraint.

ACKNOWLEDGMENTS

CLARE is being developed as part of a collaborative project involving SRI International, British Aerospace, BP Research, British Telecom, Cambridge University, the UK Defence Research Agency, and the UK Department of Trade and Industry.

REFERENCES

Abe, M., Y. Ōshima, K. Yuura and N. Takeichi (1986) "A Kana-Kanji Translation System for Non-Segmented Input Sentences Based on Syntactic and Semantic Analysis", *Proceedings of the Eleventh International Conference on Computational Linguistics*, pp 280-285.

Alshawi, H. (1990) "Resolving Quasi Logical Forms", *Computational Linguistics* 16:3, pp. 133–144.

Alshawi, H. (ed.) (1992) *The Core Language Engine*, M.I.T. Press.

van Berkel, B., and K. De Smedt (1988) "Triphone Analysis: A Combined Method for the Correction of Orthographical and Typographical Errors", *Proceedings of the Second Conference on Applied Natural Language Processing*, pp. 77–83.

Carter, D.M. (1987) "An Information-theoretic Analysis of Phonetic Dictionary Access", *Computer Speech and Language*, 2:1–11.

Carter, D.M. (1989) "Lexical Acquisition in the Core Language Engine", *Proceedings of the Fourth Conference of the European Chapter of the Association for Computational Linguistics*, pp 137–144.

Emirkanian, L., and L.H. Bouchard (1988) "Knowledge Integration in a Robust and Efficient Morpho-syntactic Analyser for French", *Proceedings of the Twelfth International Conference on Computational Linguistics*, pp 166–171.

Futrelle, R.P., C.E. Dunn, D.S. Ellis and M.J. Pescitelli, Jr. (1991) "Preprocessing and Lexicon Design for Parsing Technical Text", *Proceedings of the Second International Workshop on Parsing Technologies*, pp. 31–40.

Grosz, B. J., D. E. Appelt, P. Martin, and F. Pereira (1987). "TEAM: An Experiment in the Design of Transportable Natural-Language Interfaces". *Artificial Intelligence* 32: 173–243.

Kernighan, M.D., K.W. Church, and W.A. Gale (1990). "A Spelling Correction Program Based on a Noisy Channel Model", *Proceedings of the Thirteenth International Conference on Computational Linguistics*, pp 205–210.

Koskenniemi, K. (1983) *Two-level morphology: a general computational model for word-form recognition and production.* University of Helsinki, Department of General Linguistics, Publications, No. 11.

Pollock, J.J., and A. Zamora (1984) "Automatic Spelling Correction in Scientific and Scholarly Text", *Communications of the ACM*, 27:4, pp 358–368.

Veronis, J. (1988) "Morphosyntactic Correction in Natural Language Interfaces", *Proceedings of the Twelfth International Conference on Computational Linguistics*, pp 708–713.