Natural Language Engineering **2** (4) 331–336. Printed in the United Kingdom

© 1996 Cambridge University Press

Finite state morphology and information retrieval

KIMMO KOSKENNIEMI

Department of General Linguistics
University of Helsinki, FIN-00014, Finland
e-mail: kimmo.koskenniemi@helsinki.fi

(*Received 31 December 1996; Revised 3 February 1997*)

Abstract

A source of potential systematic errors in information retrieval is identified and discussed. These errors occur when base form reduction is applied with a (necessarily) finite dictionary. Formal methods for avoiding this error source are presented, along with some practical complexities met in its implementation.

1 Introduction

There are certain morphological tasks which are relevant to information retrieval but which do not seem important when morphological analysis is considered in general. In this paper, we study the effects of morphological analysis in a framework of full-text retrieval. We assume that a set of documents is being accessed according to individual words occurring in them. The language in the documents is assumed to be inflecting to the extent that some morphological processing is needed.

If the search were based on the keywords in their base form, and the index were to contain the unprocessed wordforms, lots of the occurrences of the keywords would be lost. Plain truncation helps, but does not solve the whole problem in all cases. Truncation is virtually useless if inflectional elements are prefixed, as is the case in Semitic languages (such as Arabic) or in Bantu languages (such as Swahili). Even in apparently easier languages, with suffixation alone, the word stems may be subject to alternations, making truncation inefficient.

2 Search stems

Document retrieval in languages which are exclusively suffixing can be approached by using so called *search stems*. This method compensates for the possible stem alternations by producing a set of prefixes which cover all possible inflected forms of the lexeme. This is done without specific reference to a dictionary, i.e. based on the productive rules of the language, e.g. the search stem generator for a Finnish would give the following results out of a noun *kauppa* 'shop': *kauppa-*, *kaupa-*, *kauppo-*, and

https://doi.org/10.1017/S1351324997001587 Published online by Cambridge University Press

kaupo-. It has proven to be possible to write such search stem generators for several languages, and some of them have been in production use for more than ten years Koskenniemi 1985.

The use of this method has no effect on the number of distinct word forms in the index file and monotonically improves recall. It fails, however, to retrieve compounds where the key word is not initial.

With inflecting languages such as Finnish, the index file (also called the inverted file) tends to grow linearly with the size of the text database. With English text databases, it is said that the number of different word forms stops growing at a certain point. No similar saturation has been observed with Finnish texts. Thus, this method is bound to be less than optimal in terms of the space requirements.

3 Analysis without a lexicon

The morphological processing could, in principle, be done without a lexicon, i.e. based exclusively on rules of inflection, so that instead of the set of actually possible base forms, one produces a larger set of base forms where all potential and theoretical possibilities are given as well, e.g. in Swedish, such an analyser without a lexicon would be forced to analyse a wordform *pojkar* at least in four ways: *"pojke" N UTR INDEF PL NOM*, *"pojka" V ACT PRES*, *"pojka" N UTR INDEF SG NOM*, and *"pojkar" N UTR INDEF SG NOM*. Only the first of these is a real alternative, others are nonexistent.

We will elaborate this method briefly in this paper from another perspective, but it suffices to note at this stage that morphological analysis without a lexicon is likely to produce lots of spurious base forms. This would be undesirable and inefficient in terms of space requirements. Such a method would possibly also be suboptimal in terms of the execution speed when retrieving documents.

4 Analysis with a lexicon

It is more common to perform the base form reduction using a fixed lexicon. For a given word form, exactly those base forms will be produced which the lexicon permits either by having an appropriate entry through a mechanism for compounding or deriving the complex stem. Thus the set of possible base forms are strictly defined and restricted in advance. This is an advantage when we consider the bulk of the wordforms in ordinary documents which are, indeed, covered by conventional and electronic dictionaries.

It is well known that the lexical inventory of normal unrestricted documents is neither fixed nor bounded — we encounter more and more proper names, new terms and acronyms as we extend our collections of texts. This problem is routinely approached by so called *morphological heuristics* or *morphological guessing algorithms* which try to deduce the possible base forms of the wordforms which could not be recognized by the main morphological analyser (see, for instance, the ENGCG parser of English Voutilainen 1995). The main difference between such heuristics and morphological analysis without a lexicon is that heuristic algorithms ignore

https://doi.org/10.1017/S1351324997001587 Published online by Cambridge University Press

the exceptional and closed inflectional classes covered by the main lexicon, and concentrate on describing the inflection of names, foreign loan words, newly coined words, etc.

5 The problem

One problem remains even after we have implemented a morphological analyser and augmented it with a morphological heuristics module. The problem arises from the combinatorics: some word forms which receive analyses and base forms from the morphological analyser might actually have as their true interpretation another lexeme, not in the lexicon.

Proper names can have almost arbitrary shapes, and we are currently exposed to news and texts from all parts of the world. Trade marks and various acronyms are being invented constantly. As an example, a Russian politician, Mr. Boris Pankin, was in the news for some time, and his name happens to be homographic to a Finnish noun *pankki* ('bank') in genitive singular. If we would be using the wordform reduction into base forms using any reasonable lexicon of Finnish, some inflectional forms (including the most common one, the nominative singular) would be associated with a wrong keyword *pankki*. As a result, these instances would not be retrieved with a query concerning *Pankin*. The same thing happens through compounding, e.g. in Finnish the word form *autonomia* 'autonomy' could be either *"autonomia" Noun Nominative Singular* or *"auton_oma" Noun Genitive Singular + Adjective Positive Partitive Plural*. The second analysis ('car' Genitive + 'owned by' Partitive) is nonsense but it would be the only analysis if there were no entry for 'autonomy' in the dictionary.

It is easy to see that it is rather hard to do anything about this particular problem in advance, i.e. while indexing the documents. We ought to suspect every single word form in the input, and it is not easy to see how the problem instances could be identified. But during the actual queries we can easily detect whether a keyword was one of the known ones, simply by applying the same morphological analyser to the keyword. If it is analysed as a base form of itself, then it is a known word, and otherwise it is not. Unfortunately, this seems to be a bit too late because the indexing has already been completed, and has made the error, because of the lack of this knowledge.

The essential question during the retrieval is: under which base forms did our lexicon put the possible inflectional forms of the given keyword. In other words, we should *find all base forms in our lexicon which have inflectional forms in common with the given keyword*.

There is a known, straightforward solution to circumvent this problem by using two index files, one containing the unmodified word forms, and the other consisting of the base forms produced using a fixed lexicon. In this scheme, the known query words are retrieved through the base form index file, and the unknown keywords through the word form index file (using a search stems). This approach is not discussed here any further because it is obviously uneconomic in terms of its space requirements.

https://doi.org/10.1017/S1351324997001587 Published online by Cambridge University Press

6 Finite state representation of a fixed lexicon transducer

Morphological analysis with a lexicon has been described in terms of finite state machines and regular relations for some time. Kaplan and Kay proposed and implemented a scheme of cascaded finite state transducers in the 1980s Kaplan and Kay 1994. Koskenniemi designed his two-level model based on this work Koskenniemi 1983; Koskenniemi 1986; Karttunen et al. 1987, and there is a host of full scale morphological analysers available now, based on this framework. Later on Karttunen generalized this scheme into a model where more than two levels could be permitted, and where the lexicon is implemented directly as a bidirectional finite state transducer Karttunen and Beesley 1992; Karttunen et al. 1992. Their approach is followed here as the formal framework of the representation.

The morphological analyser using a fixed lexicon can be expressed in terms of a finite state transducer, or a regular relation R. This relation is not one-to-one because wordforms may be ambiguous. It is not defined over all strings of the alphabet either, since ungrammatical wordforms have no analysis. We study here a special version R_L of this relation: one which associates pairs of strings consisting of a wordform ω which is a string over the alphabet Σ and a base form λ (which is a string over the alphabet Σ) extended with the part of speech code γ (which is a one symbol long string of Γ). We thus have $(\lambda\gamma)R_L(\omega)$ if and only if ω is a valid inflectional form of the base form λ which has the part of speech γ.

7 Lexicon transducer for morphological heuristics

We will make use of a transducer for the open lexicon (morphological heuristics/guessing) which we call R_H. We try to sketch the definition this transducer. First, we replace the fixed list of stems with a general expression which is open enough to cover all anticipated proper names, foreign words, new terms, acronyms etc. which could occur in texts. It is worth noting that only productive inflectional patterns need to be included in R_H, since exceptions and idiosyncratic patterns are not likely to occur in new words.

For certain types of languages, it is not particularly difficult to express R_H if we already have a formulation of R_L at hand. If no stem variations are associated with the inflectional class, then an expression such as $\Pi_0 = \{(\sigma : \sigma)^* : \sigma \in \Sigma\}$ could be inserted in the lexicon (and at the same time, all existing entries in that class could be removed).

If stem final alternations occur in a class, then the Π_0 inserted in place of the (truncated) stems to be followed by the existing set of alternating stem ends (a so called minilexicon) handling the alternation, and these stem ends, in turn would be followed by the actual inflectional endings, as in the standard lexicon.

If morphophonemes are anticipated to occur in the stems, they have to be properly embedded in the expressions and surrounded by 'wild cards', and properly chained to sets of inflectional endings. We tentatively assume that a relation R_H can be expressed following such lines, and using, for example, the LEXC formalism (see Karttunen 1993 for more details).

https://doi.org/10.1017/S1351324997001587 Published online by Cambridge University Press

There is one serious reservation to this simple scheme. The regular expression used as the building block in the above construction is of the type $Y = \Sigma^* X$ where X is an expression corresponding to a sizable finite state machine. The deterministic version of automaton corresponding to Y tends to get (exponentially) large. Thus we cannot claim that the concrete compilation of the run-time R_H is feasible.

8 A theoretical solution

These two transducers, if available, would provide us with an answer to the main question presented above. Let us suppose that a given keyword $\lambda\gamma$ is not defined by the fixed lexicon. If we would produce the (possibly very large) set of wordforms ω through the transducer R_H^{-1} and analyse them with the transducer R_L we would get the set of base forms $\lambda_1\gamma_1, \ldots, \lambda_k\gamma_k$ for which some inflectional forms coincide with some inflectional forms of the $\lambda\gamma$.

The transducers involved here are constant, not depending on the keyword $\lambda\gamma$. Thus we could avoid the (often impossible) task sketched above by computing the transducer corresponding to the composition $R_C = R_L \circ R_H^{-1}$. This relation $(\lambda_i\gamma_i)R_C(\lambda\gamma)$ holds if and only if λ and $\lambda_i\gamma_i$ have common inflectional forms. In other words, the formula would look like:

$$(\lambda_i\gamma_i)R_C(\lambda\gamma) \Leftrightarrow \exists\omega \in \Sigma^* : ((\lambda_i\gamma_i)R_L(\omega) \wedge (\omega)R_H(\lambda\gamma))$$

R_C is expected to be more reasonable in terms of its size because it is constrained by the existing fixed lexicon R_L. However, as we noted before, R_H is likely to be too large. We conjecture that the transducer R_C could be computed more easily by reversing the components, i.e. by constructing R_H as a reversed transducer $R_H^{\leftarrow}$ which is composed with the reversal of the fixed lexicon transducer:

$$R_C = (R_L^{\leftarrow} \circ (R_H^{\leftarrow})^{-1})^{\leftarrow}$$

9 Future work

The scheme which was presented in this paper has been verified in a rather small scale using the Xerox Finite State Calculus Karttunen and Yampol 1993. It remains to be implemented with a full scale description of some example language.

We have not discussed the full complexity of dealing with unknown keywords. More work is needed, e.g. to elaborate general and comprehensive methods to construct the transducer R_H for all types of languages. Furthermore, various more complex cases, e.g. where one non-initial component of a compound is missing from the dictionary, remain to be worked out in more detail.

It is likely that there are other paths along which R_C could be built. For example, one could try refrain from determinizing R_H before the composition, as suggested by Pasi Tapanainen (p.c.). The strategy presented in this paper might work when using our current formalism and rules, but this might require some careful planning and/or preprocessing of the inflectional information in the fixed lexicon.

https://doi.org/10.1017/S1351324997001587 Published online by Cambridge University Press

References

Antworth, Evan. 1990. *PC-KIMMO: A Two-level Processor for Morphological Analysis*. Texas: Summer Institute of Linguistics.

Barton, Edward, Berwick, Robert, and Ristad, Eric. 1987. *Computational Complexity and Natural Language*. The MIT Press.

Ejerhed, Eva and Church, Kenneth. 1983. Finite state Parsing. In Karlsson, Fred (ed.), *Papers from the Seventh Scandinavian Conference of Linguistics* **9–10**: 410–432. Department of General Linguistics, University of Helsinki, Publications, Number .

Kaplan, Ronald and Kay, Martin. 1994. Regular Models of Rule Systems, *Computational Linguistics* **20.3** pp. 331–378.

Karlsson, Fred. 1990. Constraint Grammar as a Framework for Parsing Running Text, In *Papers presented for the 13th International Conference on Computational Linguistics, COLING-90* **3** pp. 168–173.

Karttunen, Lauri and Beesley, Kenneth. 1992. Two-level Rule Compiler, Report ISTL-92-2, Xerox, Palo Alto Research Center, California.

Karttunen, Lauri. 1993. Lexicon Compiler, Xerox, Palo Alto Research Center, California.

Karttunen, Lauri and Yampol, Todd. 1993. Interactive Finite state Calculus, Xerox, Palo Alto Research Center, California.

Karttunen, Lauri, Kaplan, Ronald M. and Zaenen, Annie. 1992. Two-level Morphology with Composition, In *COLING-92*, pp. 141–148.

Karttunen, Lauri, Koskenniemi, Kimmo, and Kaplan, Ronald. 1987. A Compiler for Two-level Phonological Rules, In Tools for Morphological Analysis, pp. 1–61. Center for the Study of Language and Information, Stanford University, California.

Kinnunen, Maarit. 1986. Morfologisten sääntöjen kääntäminen äärellisiksi automaateiksi. (Translating morphological rules to finite state automata), Department of Computer Science, University of Helsinki.

Koskenniemi, Kimmo. 1983 Two-level Morphology: A General Computational Model for Word-Form Recognition and Production, University of Helsinki, Department of General Linguistics, Publications, No. 11.

Koskenniemi, Kimmo. 1986. Compilation of Automata from Morphological Two-level Rules. In Karlsson, Fred (ed.) *Papers from the Fifth Scandinavian Conference on Computational Linguistics,* Helsinki, December 11–12, 1985, pp. 143–149. University of Helsinki, Department of General Linguistics, Publications, No. 15.

Koskenniemi, Kimmo. 1985. FINSTEMS: A Module for Information Retrieval, In Karlsson, Fred (ed.), *Computational Morphosyntax: Report on Research 1981–84*, pp. 81–92. University of Helsinki, Department of General Linguistics, Publications, No. 13.

Koskenniemi, Kimmo. 1990. Finite-state Parsing and Disambiguation, *Papers presented for the 13th International Conference on Computational Linguistics, COLING-90* **2**: pp. 229–323.

Koskenniemi, Kimmo and Church, Kenneth. 1988. Two-level Morphology and Finnish, *Proceedings of the 12th International Conference on Computational Linguistics, COLING-88* Budapest, pp. 335—339.

Koskenniemi, Kimmo, Tapanainen, Pasi and Voutilainen, Atro. 1992. Compiling and Using Finite-State Syntactic Rules, *Proceedings of the 14th International Conference on Computational Linguistics, COLING-92*, Vol. 1, Nantes, 156–162.

Voutilainen, Atro. 1995. Experiments with heuristics. In Karlsson, Fred, Voutilainen, Atro, Heikkilä, Juha and Anttila, Arto (eds.), *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*, pp. 293–314. Berlin, New York: Mouton de Gruyter.

https://doi.org/10.1017/S1351324997001587 Published online by Cambridge University Press