Morphemic Analysis: A Dictionary Lookup Instead of Real Analysis

Pavel Šmerk

Faculty of Informatics, Masaryk University
Botanická 68a, CZ-602 00 Brno, Czech Republic
smerk@mail.muni.cz

Abstract. This paper presents an approach for developing morphological and morphemic analysis systems for inflective languages based on a simple and fast dictionary lookup instead of any kind of analysis of the input word form. This approach allows the information about the word forms (lemma, tag, morpheme structure, derived words, derivational relations) to be described according to the traditional grammars' models and to have such a description completely independent of any requirement of the analysis process.

1 Introduction

According to Gelbukh and Sidorov [5], the designer of a morphological analyzer for an inflective language has the following choice:

- either generate all word forms and build a system with a large dictionary and a very simple "analysis" (just searching) algorithm,
- or build a system with a much smaller dictionary of stems with information about possible endings, but with some more sophisticated algorithm (analysis through generation, in particular).

For the inflective languages they strongly suggest the second option, because it allows to use a grammar model almost directly taken over from traditional grammars, which are oriented mainly toward generation. These traditional models are rather simple, but foremost intuitive for a system developer or morphological database editors.

The aim of this paper is to show that it is possible to preserve all advantages of the use of traditional grammars' models retaining quite simple "analysis" by word forms dictionary searching algorithm as well. And, moreover, this applies not only to the morphological analysis, but even to the morphemic analysis, i.e. formal description of the derivational morphology. The suggested approach is based on the use of Jan Daciuk's [2] algorithms and tools for construction of minimal deterministic acyclic finite state automata (DAFSA) and on his (but not only his, cf. further) idea of how to use such automata for morphological analysis.

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2007, pp. 77–85, 2007.

Actually, the basic ideas of all what follows were published by Jan Daciuk and other authors almost ten years ago. It is hard to believe that there does not seem to be any real world implementation in the time being, at least for Slavic languages, for which it could be very interesting due to their high inflectionality. There are some experimental implementations, of course, but not any really used Slavic inflectional or even derivative morphology based on Daciuk's DAFSAs or on any similar approach.

The following section illustrates the basic idea, why it is advantageous to represent a morphological dictionary by the minimal DAFSA. It does not get stuck into technical details concerning the actual construction of the minimal DAFSA. The algorithms are published, and even ready-to-use tools are available.[1] Instead of it, in Section 3, shows the possible arrangements of the data for morphological and morphemic analysis and guessing. Section 4 discusses some further advantages of the proposed approach.

2 Basic Idea

The basic idea is very simple, but very powerful. Any finite list of unique strings can be considered to be a finite (formal) language and as such can be represented by some DAFSA. If we choose the minimal one, then a partial path corresponding to any left or right substring shared by some subset of the modelled strings will occur exactly once in such minimal automaton.[2]

For example, let us consider the following fraction of some dictionary of the word form and morphological tag pairs[3]:

```
Kanaďánek:c1nSgMk1
Kanaďánka:c2nSgMk1
Kanaďánkovi:c3nSgMk1
Holanďánek:c1nSgMk1
Holanďánka:c2nSgMk1
Holanďánkovi:c3nSgMk1
```

One can see that all pairs with the word forms of the same lemma (lexeme) *Kanaďánek* or *Holanďánek* share some same left substring, in particular *Kanaďán* and *Holanďán* respectively. Similarly, all pairs with the nominative, genitive and dative singular of the masculine animate nouns share at least the same right substring *:c1nSgMk1*, *:c2nSgMk1* and *:c3nSgMk1* respectively — but in the case of the same declension or derivational type they can share a few characters

[1] `http://www.eti.pg.gda.pl/katedry/kiw/pracownicy/Jan.Daciuk/personal/fsa.html`

[2] To be precise, this is not true in some of the cases, when in the same string some left substring shared by more strings overlap some right substring, which is also shared by more strings. But due to the regularity of an absolute majority of words in a natural language it does not affect the following claims dramatically (cf. some results for Nonslavic languages at the beginning of the following section).

[3] All examples in this paper are in Czech: *Kanaďánek* is a little Canadian boy (diminutive), *Holanďánek* is a little Dutch boy. The text after the colon is a morphological tag: c1, c2 and c3 is the first, the second and the third case (i.e. nominative, genitive and dative), nS is number – singular, gM is gender – masculine animate, k1 is part of speech – noun.

more: in the data above they share even *ďánek:c1nSgMk1*, *ďánka:c2nSgMk1* and *ďánkovi:c3nSgMk1*. Moreover, all pairs of this example share e.g. the *nSgMk1* right substring. Since the inflective languages tend to express all morphological categories at the end of the word form (and, moreover, often in one single ending, but it is not important for now), we can take for granted that similar relations will hold through the whole list of all word form and tag pairs.

The analysis based on an automaton which encodes a list of the word form and tag pairs is quite straighforward. If an analysed string concatenated with the separator (the colon in the case above — it serves as the sign of the end of the string to distinguish for example *Kanaďánka* and *Kanaďánkama*[4]) is found in the automaton then each possible remaining path to the final state of the automaton encodes one of the possible morphological tags. The "searching" of the string is as simple (and therefore also quick, of course) as possible, because the automaton is deterministic, so that for each possible string there is at most one straightforward path without need of any backtracking.

Apparently, the creation of such a minimal DAFSA can be a simple way to dramatically reduce the size of the list of word form and tag pairs (or any other similar list) and thus to get rid of the disadvantage of the list's big size. Moreover, the creator of the list has a full freedom of choice of the manner in which he or she creates the list. The method used will have no influence on the effectiveness or complexity of the proces of analysis, because the management of the data is completely separated from their usage.

There is only one condition which has to be met to get really compact representation of the dictionary: strings representing the data must not have any unique parts. The next section shows how to arrange the data for various parts of the morphological and morphemic analysis to meet this condition.

3 Data Representation

As was shown above, the list of word form and tag pairs is suitable for the construction of the DAFSA. But a serious problem arises when we want to include the information about the lemma:

```
Kanaďánek:Kanaďánek:c1nSgMk1
Kanaďánka:Kanaďánek:c2nSgMk1
Holanďánek:Holanďánek:c1nSgMk1
Holanďánka:Holanďánek:c2nSgMk1
```

Obviously, all combinations word+lemma and lemma+tag are unique among all strings. DAFSA can be constructed, of course, but it would be too large. The solution used by e.g. Daciuk [2], Kowaltowski [8] or in a slightly different manner by the INTEX project [10] is simple: store only the right substring of the lemma, in which it differs from the particular word form:

[4] colloquial form of the instrumental plural

```
Kanaďánek::c1nSgMk1
Kanaďánka:Bek:c2nSgMk1
Holanďánek::c1nSgMk1
Holanďánka:Bek:c2nSgMk1
```

where the *B* as the second letter of the aplhabet means “to get the lemma delete two last characters from the word form and then attach the *ek*". It is clear now, that the list in this form has the same properties as the list of word form and tag pairs in previous section and as such can be “compressed" to a minimal DAFSA. The lookup is similar as was above: after finding the word form, the remaining paths to the final state describe all proper lemma+tag combinations. It should be noted that the compression rate could be very high. Kowaltowski [8] reports 0.25 byte per one word-tag-lemma entry for Brazilian Portuguese and Daciuk [2] reports less then 0.15 byte per one word-lemma-tag entry for German and ca. 30 times better compression rate compared to gzip on that data. Quite preliminary results for Czech data show similar compression potential.

The following subsections are only variations on this solution.

3.1 Generating All Word Forms from the Lemma

This is quite simple. All we need is to swap word form and lemma in order to create list of lemma:word form:tag triples. Instead of the full word form there is, of course, only an ending part in which the word form and lemma differ. The generation is similar to the above: the lemma is looked up and the remaining paths describe all possible word forms with proper tags. To generate all word forms from a word form other then lemma it is the best option to analyse the input word form to get the lemma and then look up the rest.

3.2 Morphemic Analysis

Let us suppose we have a mechanism, which can handle productive derivational suffixes and derive words according to some rules. Then we derive all word forms we can (maybe except for some recursive rules as for great-great-grandmother and so on) and track the history of the derivation. Of course we track the internal, “deep" forms of morphemes before any phonological rules or morphophonological or stem internal alternations apply. Then a dictionary may look as follows:

```
Kanaďánek:Ed-an-0k-~
Holanďánek:Ed-an-0k-~
Kanaďánka:Ed-an-0k-a
Holanďánka:Ed-an-0k-a
```

where the *E* as the fifth letter in the alphabet means “strip last five characters", as above. The first segment, *d*, serves for recovering the root

morpheme (whose last phoneme has been palatalised, so the base form is *Kanad-* as English *Canad-*), the following segments are derivational suffixes and the last one is the inflectional ending.[5]

Obviously, such a morphemic "analysis" is as simple as the morphological analysis above. It constrasts with Zeldes's [12] very recent attempt to morphological/morphemic[6] analysis of Polish resulting in the analysis algorithm which is "computationally more complex, but lexicographically more compact alternative to text-based morphological analysis techniques currently in use for Polish"

3.3 Generating All Derived Words

To generate derived words we need a list of stems with possible suffixes[7] encoded in the usual way:

```
Kanad:Aďan
Kanaďan:Bánek
Holand:Aďan
Holanďan:Bánek
```

The interpretation (generation) has to be (or can be) recursive, e.g. *Kanad-* → *Kanaďan* → *Kanaďánek*.

Another list is needed for determining the word which the analysed word was derived from:

```
Kanaďan:Cd
Kanaďánek:Dan
Holanďan:Cd
Holanďánek:Dan
```

It allows recursive interpretation as well. It is sufficient to have both these lists created for the lemmata only, not for all word forms.

3.4 Morphological and Morphemic Guessing

The dictionaries for guessing morphological or morphemic characteristics of unknown words are a bit different. There are no full word forms in them, but only the "surface" forms of endings or endings with sequences of suffixes, possibly followed by the alternated root final consonant(s). This whole potential right substring of an unknown word is reverted and the analysed words are matched from their ends. It is of a high importance to use only really productive endings and suffixes to avoid the overgeneration. The first list is for the lemma and tag guessing:

[5] *0* stands for vowel *e* alternating regularly with zero and ~ stands for zero ending — of course, all these signs are chosen completely arbitrarily. [6] Zeldes wants to get a linguistically adequate split of an analysed word to the stem and ending, he does not perform a full morphemic analysis

[7] Yes, there are also prefixes in the language, let us put them aside, as handling them would be technically more difficult, but it would not bring any new idea.

```
kenáď::c1nSgMk1
aknáď:Bek:c2nSgMk1
```

the second for the morphemic guessing:

```
kenáď:Ed-an-0k-~
aknáď:Ed-an-0k-a
```

The structures of both are the same as in the previous subsections.

This paper has started with a polemic against Gelbukh and Sidorov [5]. The same two argued a year before [4] that (only) the algorithmic, non-dictionary approaches to the morphological analysis allow the guessing of the unknown word forms. It should be clear now, that even "dictionary" approach allows the same.

4 Advantages of the Proposed Approach

There are certainly no doubts that natural language processing needs to have a usable and reliable derivational (and *really* derivational, e.g. not only static description of derivational relations between lexicon entries) morphology.

Then the main advantage of the proposed approach arises from the fact, that one has to describe and implement all morphophonological alternations or at least large part of them to accomplish really good derivational morphology (and morphemic analysis as well). The "every exception is a new paradigm"-like approaches (e.g. for Czech language analyser `ajka` [9]) or Jan Hajič's analyser [6]) hardly can be successful, because they are too redundant from the derivation's point of view, and therefore too complex regarding the maintainance of the data. Consider, for example, that a need of addition of some colloquial noun endings appears: having several hundreds noun paradigms, it will lead either in a creation of some ad hoc and thus possibly erroneous scripts, or in an manual update of the most frequent paradigms only. Both cases can cause inconsistencies in the data[8].

On the other hand, a system with a small number of paradigms which computes[9] these morphophonological (or even stem or root internal, if someone wants to distinguish it) alternations in the realtime (i.e. during the analysis phase) has and has to have too complex code to maintain as well. Or, such a system is too complex at least compared to the approach proposed in this paper, which is: however complex description of the data you have or want to have, for the analysis generate all of them in advance and then use only simple static dictionaries.

Even the well known two-level morphology [7] leads to superfluous complexity and unevincible linguistic inadequateness (comparing to described approach) when used for Slavic languages. Either we have to have a list of stem alternations (which are rather frequent in these languages). But such a solution

[8] This is one of the real present-day deficiencies of the `ajka` analyser. [9] This "computes" means some kind of algorithm more complex than a simple walk through some FSA or FST.

resigns to any adequate description of the regularity of these alternations. Or we can encode these alternations into the phonological rules which are likely to be linguistically inadequate. And if someone wants to use some prepared tools like Xerox's `xfst` [1][10], either has to develop some translation from his or her data description to the formal language that the particular tool uses, or is bound to that language.[11]

Thus, the main advantage is the complete separation of the "analysis" process and the description of the data. This separation allows not only the free choice of the model for morphological and morphemic description of the data (namely of the possible morpheme combinations and the alternations caused by these combinations), but it also to a great extent simplyfies changes of this description when needed.

Very simple example: let us suppose some set of words which belong to a particular paradigm, and another set of words which all have some same additional form (e.g. due to some diachronnic reasons), so that the analyzer distinguishes two separate paradigms differing only by this one extra form. Now let us imagine, that we would like to lower the redundancy of our paradigm system by employing some inheritance principles. We would want to have one base paradigm and one derived from this base one by adding the extra form. To achieve this in traditional models of analysis[12], we would have to not only modify the tools managing the database of lexemes and paradigms, but also the analyzer. And the second could be not so simple as the analyzers are in general optimized for high speed of processing and low size of the data. Using the approach proposed in this paper, the proper modification of the database managing tools suffices in such case and the analyzer itself may remain intact.

4.1 Description of Productivity

The described approach is very handy especially for the description of the regularities in the language, namely the process of the derivation of new or infrequent words. For instance, most of newly created or taken over Czech verbs adopt the suffix *-ov* so that the lemma ends with *-ovat*, e.g. *programovat*[13]. But there is a whole bunch of suffixes completely regularly connected with this type of verbs: *-ován*, *-ovaný*, *-ování*, *-ovaně*, *-ovanost*, *-ovatelný*, *-ovatelně*, *-ovatelnost*[14]

Adding a verb from this class to a dictionary of word form, lemma and tag triples adds in the DAFSA only the stem or some its part and one arc to the common beginning of all these suffixes. But the language of the DAFSA is by such an addition enriched with all word forms generated from lemmas, which means about 150 word forms (many of them are homonymous, however). Of

[10] But unfortunately `xfst` is not freely available. [11] Of course, it is possible to process some part of derivational morphology in this way, as [11] did for Czech. In her paper only three rules are shown — and all of them are awkward if one imagine that whole derivational morphology should be done in this "write-only" manner.
[12] at least in all really used ones we are aware of [13] to programme [14] Taking the verb to programme, the glosses can be something like: programmed (pass part), programmed (adj), programming, ?, ?, programmability, programmable (adj), programmable(?) (adv).

course, it would be useful to have some mechanism of (some tool for) an simple adding of a new word to such a class also on the generating side of the analyzer system, i.e. the suffixes should be really interconnected somehow. However, in this is the power of this approach: one arc in the automaton is able to enrich the accepted language with even thousands of words. On the other hand, many of these c. 150 words will occur in real texts rarely or even never. And many of them or may be even all of them could be correctly analysed as unknown words. But why try to guess if we are able to know, and it worths almost nothing? On the other hand, even if we prefer guessing in some cases, it can be very simplified using the described approach.

4.2 Some Minor Advantages

- The description of morphophonological alternations needs not to be efficient, because it does not affect the process of analysis at all. It is very important, as it allows to use the description, which is really adequate for the data. Moreover, it allows the free choice of the programming language for tools managing the data, e.g. some high level and more comfortable scripting language. It also allows the scripts (or programs) to be optimized for maintainability, which is important for long-life projects.
- It allows to have either several smaller cooperating tools or several quite independent parts of a analyzer, thus the whole project would be less complex, which may prevent some programmers' mistakes from arising.
- This approach also allows a gradual move from some previous morphological analysis system. It is possible to have some parts of the derivational (or, of course, also inflective) morphology described in the new system with proper capture of important productive relations, and to take over the rest (perhaps non-productive, and therefore more tricky to describe) of the morfology from the previous system in a form of a plain list of word form, lemma and tag triples etc.

5 Conclusion

As was said at the beginning, the basic ideas of the proposed approach are not new at all, but their potential seem to be rather underestimated and maybe they worth to be recalled a bit, what was the aim of this paper.

A new morphological and morphemic analyser for Czech is developed using this approach, but still in the phase of data preparation (unification of related paradigms etc.), therefore there are no results available yet.

Acknowledgments

This work has been partly supported by the Academy of Sciences of Czech Republic under the project 1ET200610406, by the Ministry of Education of CR within the Center of basic research LC536 and by the Czech Science Foundation under the project 201/05/2781.

References

1. Kenneth R. Beesley and Lauri Karttunen. 2003. Finite State Morphology. CSLI Publications, Stanford University. 509 p.
2. Jan Daciuk. 1998. *Incremental Construction of Finite-State Automata and Transducers, and their Use in the Natural Language Processing.* Ph.D. dissertation, Technical University of Gdańsk, Poland.
3. Jan Daciuk. 2001. *Experiments with Automata Compression.* Proceedings of Conference on Implementation and Application of Automata CIAA'2000. LNCS 2088, Springer-Verlag, pp. 105–112.
4. Alexander Gelbukh and Grigori Sidorov. 2002. *Morphological Analysis of Inflective Languages through Generation.* In: J. Procesamiento de Lenguaje Natural, No 29. Sociedad Española para el Procesamiento de Lenguaje Natural, ISSN 1135-5948, pp. 105–112.
5. Alexander Gelbukh and Grigori Sidorov. 2003. *Approach to Construction of Automatic Morphological Analysis Systems for Inflective Languages with Little Effort.* In: Computational Linguistics and Intelligent Text Processing. Proc. CICLing-2003. LNCS 2588, Springer-Verlag, pp. 215–220.
6. Jan Hajič. 2004. Disambiguation of Rich Inflection: Computational Morphology of Czech. Charles University, The Karolinum Press. 328 p.
7. Kimmo Koskenniemi. 1984. *A General Computational Model for Word-Form Recognition and Production* Proceedings of COLING-84, Stanford University, pp. 178–181.
8. Tomasz Kowaltowski, Cláudio L. Lucchesi, and Jorge Stolfi. 1998. *Finite Automata and Efficient Lexicon Implementation.* Technical Report IC-98-02, University of Campinas, São Paulo.
9. Radek Sedláček and Pavel Smrž. 2001. *A New Czech Morphological Analyser* `ajka`. In Proceedings of the 4th International Conference TSD 2001. LNCS 2166, Springer-Verlag, pp. 100–107.
10. Max Silberztein. 1998. *INTEX 4.1 for Windows: A Walkthrough.* Proceedings of The Third International Workshop on Implementing Automata, WIA'98. LNCS 1660, Springer-Verlag, pp. 230–243.
11. Hana Skoumalová 1997 *A Czech Morphological Lexicon* Proceedings of the Third Meeting of the ACL Special Interest Group in Computational Phonology, pp. 41–47.
12. Amir Zeldes. 2006. *Abstracting Suffixes: A Morphophonemic Approach to Polish* Proceedings of KONVENS 2006 (Konferenz zur Verarbeitung natürlicher Sprache), Universität Konstanz.