

Tools for Fast Morphological Analysis Based on Finite State Automata

Pavel Šmerk

Natural Language Processing Centre
Faculty of Informatics
Masaryk University

6. 12. 2014

Introduction

- new implementation of (some of) Jan Daciuk's algorithms and tools for morphological analysis based on finite state automata
- in particular reimplemented version of
 - tool which builds the automata from an input set of strings
 - tool which performs the morphological analysis itself
- both tools are faster and with significantly smaller/simpler code
- + unicode-aware versions

Data for Morphological Analysis

- list of strings query:answer
- pairs of words are encoded as the first word and the difference

ježek:A:k1gMnSc1 ← ježek:ježek:k1gMnSc1

ježka:Cek:k1gMnSc2 ← ježka:ježek:k1gMnSc2

ježka:Cek:k1gMnSc4 ← ježka:ježek:k1gMnSc4

krtek:A:k1gMnSc1 ← krtek:krtek:k1gMnSc1

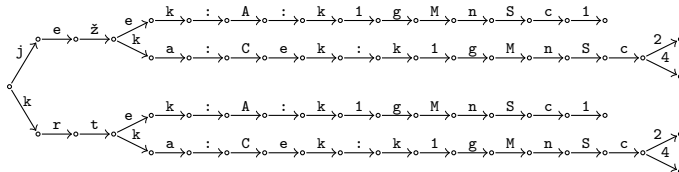
krtka:Cek:k1gMnSc2 ← krtka:krtek:k1gMnSc2

krtka:Cek:k1gMnSc4 ← krtka:krtek:k1gMnSc4

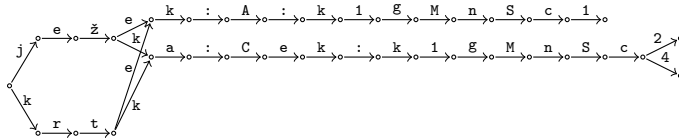
- raw list is too big, but taken as an formal language, we can construct minimal deterministic FSA of feasible size
- incremental minimization algorithm, where the FSA is minimal during construction, was invented by Jan Daciuk

Data for Morphological Analysis

- deterministic FSA



- deterministic FSA after minimization



- “analysis” is only fast and simple pass through this FSA
 - deterministic pass through the FSA according to the “query”
 - and recursive retrieval of all possible “answers”

Experiments

- data sets

data set	input size	words (lines)	output size
EN	1,417,920	88,652	244,764
RU	114,605,988	2,844,516	3,639,960
CZ free	105,001,670	3,393,080	931,594
CZ full	828,973,970	27,764,093	3,795,423

- build times in seconds for the three compared tools

data set	fsa <u>build</u>	morfologik	new implem.
EN	0.24	0.59	0.08
CZ free	12.63	7.50	4.19
RU	26.04	10.19	9.41
CZ full	121.41	57.21	41.71

- morfologik is java implementation of the builder (Dawid Weiss)

Size of Code

- another advantage of the new implementation is the size of code
- Daciuk's implementation includes many compile time options, but even after “unifdeffing” the code remains rather complicated
- files needed for the builder
 - Daciuk: 1250 lines, 35 kB
 - new code: 250 lines, 10 kB
- files needed for the analyser
 - Daciuk: 900 lines, 25 kB
 - new code: 230 lines, 8 kB
- it's easier to improve the more simple code
 - we change the edges to have labels of variable size which allows for utf-8 labelled edges
 - $\Rightarrow$ the analyzer can easily do the case conversions or diacritics restoration

Future Work

- the new tools are ready to use, but it is still a work in progress :-)
- we want to reduce
 - compile time: simple hash instead of Daciuk's "treetable"
 - run space: VLEncoded information, relative addresses, ...
 - run time: smaller run space
- for unicode versions, codepoint may perform better than utf-8
- for the morphological analysis, tags or may be whole "answers" could be stored directly in memory