Finite State Morphology Tool for Latvian

Daiga Deksne
Tilde
Vienības gatve 75a, Riga, Latvia
daiga.deksne@tilde.lv

Abstract

The existing Latvian morphological analyzer was developed more than ten years ago. Its main weaknesses are: low processing speed when processing a large text corpus, complexity of adding new entries to the lexical data base, and limitations for usage on different operational platforms. This paper describes the creation of a new Latvian morphology tool. The tool has the capability to return lemma and morphological analysis for a given word form; it can generate the required word form if lemma and form description is given; it can also generate all possible word forms for a given lemma. As Finite state transducer (FST) technology is used for the morphology tool, it is easy to extend the lexicon, the tool can be reused on different platforms and it has good performance indicators.

Introduction

An efficient way to generate forms and obtain morphological information about a word in a text is to apply morphological analysis tools. Such tools and their efficient implementation are especially important for languages that have a rich morphology. In this paper, we describe a new morphological processing tool for the Latvian language.

The Latvian language is an inflectional language. As described in (Skadiņa et al., 2012), words change form according to grammatical function. Most word forms are built by adding an affix to the stem of the word. The endings are ambiguous. The same lexical ending can symbolize several grammatical word forms. There can also be changes in a stem – regular consonant changes at the end of a stem, or a stem can be completely different for a word form. For example, for the verbs of the first conjugation, the full set of inflectional word forms is generated using three different stems - infinitive, present tense, and past tense stems. To describe the morphological lexicon, the relationships between stems and different affixes must be defined.

The existing Latvian morphological analyzer was developed more than ten years ago. It is based on a relational lexical data base of contemporary Latvian language. Prefixes, stems, and endings are stored in separate tables and are marked by different predefined declension groups. The relationship tables define eligible combinations of affixes. To be used by the morphological analyzer, this data base is compiled into a proprietary format. The same data base is used for building a spelling checker data file for Latvian. The main weaknesses of the existing Latvian morphological analyzer are: low processing speed when processing a large text corpus, complexity of adding new entries to the lexical data base, and limitations on platforms (it works only on the Windows platform). These factors promoted the search for a new solution.

In next chapters, we describe in detail the proposed solution.

1 Substantiation for chosen architecture

Existing morphological analysis tools for different languages and ways to describe the morphological lexicon were analyzed. There are many morphology tools for different languages that use FST technology. A good example from which to borrow ideas about morphologically tagged lexicon representation in finite state format is

SMOR (Schmid et al., 2004) – a morphological analyzer for German. Concatenation is used to concatenate previously defined prefixes, stems, suffixes, and inflectional endings. As these parts are marked with agreement features, filters are applied to eliminate invalid sequences.

There are several toolkits available which help in developing FST based solutions: Stuttgart Finite-State Transducer Tools[1], OpenFst library[2], Foma finite state library[3], Helsinki Finite-State Transducer Technology toolkit[4], Lttoolbox[5].

To describe the lexicon, we use the Stuttgart Finite-State Transducer Toolkit (SFST) as its extended regular expressions based transducer specification language allows to clearly describe the lexicon, to define variables, to apply concatenation, composition, insertion, and other operators needed for transducer implementation (Schmid, 2005). For transducer compilation, we use OpenFst as it supports weighted finite state transducers, and its source code can be compiled on Linux and Windows platforms. Examples in text are presented in SFST syntax.

2 Designing finite state morphology tool

There are three different transducers in the morphology tool: morphological analysis, form synthesis, and all word form generation for a given lemma. Table 1 shows the input and the output strings produced by every transducer. The files for the synthesis transducer are generated from the all form generation transducer. Only the last symbol differs in the transducer. For synthesis, the empty symbol on the input level changes to form ID on the output level (1), but for all word form generation, the form ID on the input level changes to the part-of-speech tag on the output level (2). The transducer for analysis is the inverted version of the transducer for synthesis.

(1) <>:<1397>
(2) <1397>:<m>

Analysis	
Input	piecu
Output	pieci<m0fpg000c0000000> pieci<m0mpg000c0000000>
Synthesis	
Input	pieci<m0fpd000c0000000>
Output	piecām
All forms	
Input	pieci<m>
Output	pieci<m0mpn000c0000000> piecu<m0mpg000c0000000> pieciem<m0mpd000c0000000> piecus<m0mpa000c0000000> piecos<m0mpl000c0000000> piecas<m0fpn000c0000000> piecu<m0fpg000c0000000> piecām<m0fpd000c0000000> piecas<m0fpa000c0000000> piecās<m0fpl000c0000000>

Table 1: Different transducer input and output for the numeral 'five'

For form description, the original form identifiers are used as they are in the lexical data base of contemporary Latvian language. When further compiling transducers with OpenFst, they are remapped to form descriptions which are based on a tagset developed in MUTEXT-East (Erjavec, 2011). Both (3) and (4) describe the same form – numeral, feminine, plural, genitive case, cardinal numeral.

(3) <1397>
(4) < m0fpg000c0000000>

The input for analysis is a word form, the output – one or more lemmas and form description tags. The input for synthesis is a lemma and a form description tag, and the output is one or several word forms. The input for full paradigm generation is a lemma with a part-of-speech tag, and the output is all word forms and their form descriptions.

The transducers are created incrementally. The final transducer is represented as a union of separate part-of-speech transducers.

(5) $morph$ = $Pronouns$ | $Numerals$ | $Others$ | $Adjectives$ | $Nouns$ | $Verbs$

1 http://www.ims.uni-stuttgart.de/projekte/gramotron/SOFTWARE/SFST.htm
2 http://www.openfst.org
3 http://foma.sourceforge.net/dokuwiki/doku.php
4 http://www.ling.helsinki.fi/kieliteknologia/tutkimus/hfst
5 http://wiki.apertium.org/wiki/Lttoolbox

Words belonging to a different part-of-speech are represented in a slightly different way. Words from non-inflected part-of-speech, such as conjunction, exclamation, particle, abbreviation, preposition, are represented as a lexical entry followed by one or several form IDs on the input level which changes to a part-of-speech value on the output level. All such entries are joined by union operators (6).

```
(6) $Others$ = bravo<1574>:<i> | …
```

Every numeral and pronoun is represented as an inflected form on the input level and a lemma on the output level followed by one or several form IDs on the input level which changes to a part-of-speech value on the output level. All such entries are joined by union operators. If the inflected form and the lemma start with the same characters, they are represented as a character sequence. In example (7), the word "man" (to me) has the lemma "es" (I), and the word "citam" (to other) has the lemma "cits" (other).

```
(7) $Pronouns$ = {man}:{es}<1474>:<p> |
    cit{am}:{s}<1634>:<p> | …
```

The ending classes, the lists of inflections, and stems are defined separately for nouns, adjectives, and verbs. In the future, the morphological lexicon will be extended mostly by adding adjective, verb, or noun stems of new words. The ending classes contain a full set of possible noun, adjective, and verb endings and will not require further changes. Before every adjective and verb ending is a prefix tag which marks with which prefix a particular ending can be used. As the nouns and verbs can have several inflected stems for a lemma, the special ending tag marks with which stem a particular ending can be used. In example (8), the ending tags are "<altEnd1>" and "<normEnd>". The ending on the output level changes to the corresponding lemma's ending. All ending groups are joined by union operator (9), and before every ending group is an ending group tag which will be used in a filter for filtering out the combinations of stems and endings marked with the different ending group tags. There are 15 ending groups for adjectives, 26 ending groups for verbs and 69 ending groups for nouns.

```
(8)  $Verb2$ = \
 <normEnd><PrefOther>{sim}:{t}<643>:<v> |
 <altEnd1><PrefOther>{a}:{t}<624>:<v> | …
```

```
(9) $VInfl$ = \
<>:<Verbmodal>$Verbmodal$ |
<>:<Verb2>$Verb2$ |
<>:<Verb3_aam_refl>$Verb3_aam_refl$ …
```

As nouns and verbs can have several stems to form a full paradigm, it is hard to write FST transformations by hand. There is a special file format for editing – the stems of the same paradigm are on the same line. This file contains two information blocks – one about predefined ending groups (10) and the other about the actual lexical entries, stems (11). The predefined ending group block is fixed; to improve the verb's morphological analysis, the editors will make changes in the lexical entries block. In the predefined ending group block, every line contains a predefined ending group tag, the maximal number of stems for a word marked with this ending group tag, and ending tags for every stem. All verb groups have infinitive and present stems, some might also have past stem, second present stem, participle stems. In example (10) ending group '<Verb2>' requires two stems but '<Verbmodal>' – four stems. In (11) verb 'barot' of ending group '<Verb2>' has two stems but word 'iet' of ending group '<Verbmodal>' - four.

```
  (10)
<Verb2>       2 |t<normEnd> |u<altEnd1>
<Verbmodal>   4 |t<normEnd>
  |u<normEnd1><altEnd1>
  |u<normEnd2><altEnd2> |<altEnd3>
```

```
  (11)
  <Verb2>             baro|t baroj|u
  <Verbmodal>         ie|t ej|u gāj|u iet|
```

The script changes lines to SFST representation and adds required tags for every stem (12).

```
  (12)
  ie<normEnd><Verbmodal>
  ej<normEnd1><Verbmodal>
  e:ij:e<altEnd1><Verbmodal>
  gāj<normEnd2><Verbmodal>
  g:iā:ej:<><altEnd2><Verbmodal>
  iet:<><altEnd3><Verbmodal>
```

Verbs and adjectives are represented as a concatenation of prefixes, stems, and endings, and nouns are represented as a concatenation of stems and endings.

```
(13)
$Verbs$ = $VPrefix$ $VStems$ $VInfl$
$Adjectives$ = $APrefix$ $AStems$ $AInfl$
$Nouns$ = $NStems$ $NInfl$
```

At this stage, every verb consists of three parts – prefix part, stem part, and ending part (14).

```
(14)
<PrefJa>jā
lob<>:ī<altEnd1><Verb3>
<Verb3><altEnd1><PrefJa>{a}:{t}<649>:<v>
```

The wrong forms are filtered out by composing transducers with filters which accept the same prefix, ending, and ending group tags between word constituents.

3 Evaluation

We evaluated 37 964 words from the Latvian part of the Latvian-English dictionary (Veisbergs, 2005). These words were morphologically analyzed on a computer with Windows 7 operating system (Intel® Core™ i7-2600 3.40 GHz processor, 8 GB RAM). The existing morphology tool spent 2 minutes on this task, e.g., 316 words per second, while the FST based morphology tool completed it in 4 seconds, e.g., 9491 words per second. The similar performance speed for this task also on a computer with Ubuntu GNU/Linux operating system (Intel® Core™ 2 CPU 6300 1.86 GHz processor, 7.74 GB RAM). The outputs of the two systems slightly differ as some errors in lexicon where fixed. The functionality of all form generation for a given lemma and part of speech was tested on the same data. First the word was analyzed, then the lemma and part of speech were extracted from the analysis output and passed on to the form generation transducer. The existing morphology tool spent 7 minutes and 25 seconds on this task, while the FST based morphology tool – 27 seconds. The speed of the all form generation functionality should be viewed only as a comparison between the previous and the new FST morphology tools as extra tasks are performed by the script while processing analysis results.

4 Conclusion and future work

In comparison with the existing morphology tool, FST technology is the better choice for morphology tool development. The new solution is faster; it works not only on Windows, but also on the Linux platform; it makes it easy to add new stems to predefined declension groups.

For now, all stems are listed in the transducer, except for nouns with suffixes '–tāj', '–um', '–ēj', '–šan', which are derived from verb stems. Future work will be to reduce the size of the lexicon by generating stems that have regular consonant changes. This task is not simple. Phonological changes occur only for words of certain inflectional classes and only in certain grammatical forms. For example, for nouns of the fifth and the sixth declension in plural genitive and for nouns of the second declension in singular genitive and all plural cases stem's last or two last consonants change to one or two different consonants (15). However, some stems do not follow this pattern.

```
(15)
$nounAlt$ = ({b}:{bj} | {c}:{č} | {d}:{ž} |
{dz}:{dž} | {l}:{ļ} | {ln}:{ļņ} | {m}:{mj} |
{n}:{ņ} | {p}:{pj} | {s}:{š} | {sl}:{šļ} |
{sn}:{šņ} | {t}:{š} | {v}:{vj} | {z}:{ž} |
{zl}:{žļ} | {zn}:{žņ} | {ll}:{ļļ} | {nn}:{ņņ})
```

Not all words are listed in the lexicon. Time by time new words are introduced into a language. Mostly these are foreign words and domain-specific terms, and many are formed as compounds. Compounding rules should be added to the transducer, which will increase its coverage.

Acknowledgments

The research leading to these results has received funding from the research project "Information and Communication Technology Competence Center" of EU Structural funds, contract nr. L-KC-11-0003 signed between ICT Competence Centre and Investment and Development Agency of Latvia, Research No. 2.8 "Research of automatic methods for text structural analysis".

References

Erjavec T. 2012. MULTEXT-East: Morphosyntactic Resources for Central and Eastern European Languages. *Language Resources and Evaluation*, 46/1, pp. 131-142.

Skadiņa I., Veisbergs A., Vasiļjevs A., Gornostaja T., Keiša I., Rudzīte A. 2012. http://www.meta-net.eu/whitepapers/e-book/latvian.pdf. Springer.

Schmid. H. 2005. A Programming Language for Finite State Transducers. *Proceedings of the 5th International Workshop on Finite State Methods in Natural Language Processing (FSMNLP 2005)*, Helsinki, Finland.

Schmid H., Fitschen A. and Heid U. 2004. SMOR: A German Computational Morphology Covering Derivation, Composition, and Inflection. *Proceedings of the IVth International Conference on Language Resources and Evaluation (LREC 2004)*, p. 1263-1266, Lisbon, Portugal.

Veisbergs A. 2005. Jaunā latviešu-angļu vārdnīca = the new Latvian-English dictionary. Rīga : Zvaigzne ABC, c2005