

■ Indexing old literary Finnish text

KIMMO KOSKENNIEMI

University of Helsinki
kimmo.koskenniemi@helsinki.fi

PIRKKO KUUTTI

Institute for the Languages of Finland
pirkko.kuutti@kotus.fi

KEYWORDS

two-level morphology
Old Literary Finnish
historical texts
finite-state transducers
HFST

ABSTRACT

A purpose of this study was to test the Helsinki Finite-State Transducer (HFST) technology tools, including its hfst-twolc compiler, the use of weighted finite-state transducers, to use the HFST tools out of Python scripts, and to use them together for comparing two related language forms. A strict procedure was followed in constructing, testing and revising two-level rules which relate written Modern Standard Finnish and Old Literary Finnish as used in the 17th century Bible. In particular, the advantages of the strict independence of the two-level rules were utilised. No practical production system was planned, but the results could be quite useful for indexing and concordancing similar Old Literary Finnish texts.

1. Corpus

A corpus of readily available old Finnish texts was needed for the study, more specifically texts whose language was sufficiently different from Modern Standard Finnish (MSF), but where the variation within the corpus was reasonable. The Finnish language used between the years 1540 and 1820 which is called Old Literary Finnish (OLF, in Finnish *vanha kirjasuomi*) is sufficiently distinct from MSF for the purposes of this study. Morphological analysers for MSF cannot be used as such for any OLF texts. The differences are greater the further one goes back in time.

The Finnish translation of the Bible from 1642 (often called *Biblia*) seemed suitable for the purposes of this project. Its language is homogenous enough and the text of *Biblia* is available as a digital text from the

Kaino¹ service of The Centre for Languages in Finland (Kotus). The whole translation consists of some 900,000 word tokens. For the present study, the fourth part of the Old Testament (VT-4, some 20,000 word tokens) and the first part from the New Testament (UT-1, some 12,000 word tokens) were selected and used together as our corpus.² A smaller corpus could have been sufficient for the design of the rules, but one needed a fair amount of text in order to extract a list of common word forms.

The material chosen was fairly but not fully homogeneous. Orthographic conventions used in the corpus were reasonably consistent, although they represent significantly more variation than what one finds in MSF texts. Some older materials might have been harder to handle, and some more recent materials might have been easier but less interesting to process.

An extract of the 1642 translation of the Bible (B1-Jes-3:11–3:12), along with its modern translation³ (1992), is given in Table 1 with some notes on the structural differences between them.

Table 1: A small extract of Biblia 1642 and the same passage in modern translation

OLF	MSF	Note
Mutta woi jumalattomita: sillä he owat pahat	Voi jumalatonta! Hänen käy huonosti,	missing word PL vs. SG, different verb and construction
ja heille maxetan nijncuin he ansaidzewat.	hänelle tehdään niin kuin hän itse teki.	different verb different construction
Lapset owat minun Canssani waiwajat	Kansani valtiat ovat lapsia,	different
ja waimot wallidzewat heitä. Minun Canssani	ja naiset hallitsevat sitä. Kansani,	different verb
sinun lohduttajas häiridzewät sinun	sinun opastajasi vievät sinut harhaan,	different verb and case
ja turmelewat tien jota si- nun käymän pidäis.	he ovat hämmentäneet askel- tesi suunnan.	different verb construc- tion

¹ <http://kaino.kotus.fi/>

² The parts VT-4 and UT-1 refer to the files available in the Kaino service.

³ <http://www.evl.fi/raamattu/1992/Jes.3.html>

There are many kinds of differences between the translations. Some reflect orthographic conventions which have changed meanwhile, such as using *w* instead of *v* and sometimes a single letter *a* instead of a double *aa* for a long vowel. OLF of those days had more features from the western dialects than MSF. The language itself has also changed meanwhile and continues to change. The changes are both phonological and morphological: the OLF texts often omit word final letters. The use of ending allomorphs was then quite different and has changed significantly even during the last fifty years, as one can see by comparing *Nykysuomen sanakirja* (Sadeniemi 1951–1961) and *Kielitoimiston sanakirja* (Grönros & Kotimaisten kielten tutkimuskeskus 2006).

One can also find some words in the corpus that are not used in the MSF, and some familiar words are used in another sense. The present study did not try to solve such problems which concern the vocabulary and the lexicon. In this study, only phonological, morphophonological and to some extent the allomorphic differences are addressed.

2. Representative example words

It is obvious from the above examples that one cannot align the Biblia 1642 with the modern translation word by word directly, because the translations are so far apart from each other. Instead of statistical word alignment and large sets of words, we use a fairly small set of carefully chosen good quality examples.

We started from the list of all word forms which occurred at least six times in the corpus. The list was browsed and some 180 example words were picked. The words were chosen so that there were a few examples from each type of systematic differences between OLF and MSF written forms of the Finnish language. Figure 1 (overleaf) shows a fragment of the word forms.

It was important that the list of example words would cover all common systematic differences between the MSF and the OLF forms, including orthographic and morphophonological ones.

caupungihin
 caupungijn
 corwes
 cuckoi
 cuolemaan
 cuoleman
 cuolluitten
 cuulitta
 cuulcan
 kärsimän

Figure 1: Some of the selected example words from the corpus

3. Example word pairs

The next step was to associate each of these OLF word forms with its likely MSF counterparts. The possible MSF forms corresponding to each OLF form were added, see Figure 2. If the OLF word form could correspond to several MSF word forms, the OLF form was repeated, see **cuoleman** and **kärsimän** below. The relation between the OLF forms and the MSF forms is inherently many-to-many, i.e., one modern form may correspond to several different old forms, and an old form may correspond to several distinct modern forms. Rules must permit some variation but still constrain the possibilities to a minimum.

kaupunkiin:caupungihin
 kaupunkiin:caupungijn
 korvessa:corwes
 kukko:cuckoi
 kuolemaan:cuolemaan
 kuoleman:cuoleman
 kuolemaan:cuoleman
 kuolleitten:cuolluitten
 kuulitte:cuulitta
 kuulloon:cuulcan
 kärsimän:kärsimän
 kärsimään:kärsimän

Figure 2: The selected OLF example word forms with their corresponding forms in MSF. The MSF form is to the left of the colon and the OLF form to the right.

4. Character by character alignment

The above example word pairs are not usable for our purposes as such, because the OLF and the MSF word forms are sometimes of different length. The OLF form often omits a final vowel, reduces long vowels into short ones and shortens geminate consonants, but sometimes geminates a consonant or adds a vowel etc.

Therefore, we must add some zero symbols as necessary so that the similar letters correspond to each other, first letter in the MSF word to the first letter in the OLF word etc. If the MSF word is longer than the OLF word, one must add one or more zeros to the old word in order to make the letters correspond to each other. Correspondingly, if the modern word is shorter, the zeros have to be added in it. The goal is that the letters in all corresponding positions would become similar. Zeros are added as necessary, but sparingly, e.g., as in *kärsimään:kärsimäØn* ‘to suffer’.

It must be stressed that a real character is used as a zero instead of an epsilon, an empty string or its representation 0 in the XFST regular expression language. For practical reasons, the Danish Ø was chosen as the zero symbol in rules and examples consistently in this article.⁴

The exact positions of the inserted zeros are as important as is the selection of the examples. The positions of the zeros determine what kinds of character correspondences we have. One must describe each correspondence with a rule, so the grammar may change a lot by changing the positions of the zeros a bit. In particular, any poorly positioned zero would force us to write more rules, and possibly very inadequate rules. The proper alignment also affects how well the grammar can apply to the rest of the corpus.

Letters representing similar (or identical) sounds ought to be matched with each other. Matching very different ones, e.g., consonants with vowels, must be avoided.

The initial insertion of the zeros was made manually using one’s linguistic intuition as a guideline. Once the zeros were in place, we converted the pairs of words into sequences of pairs⁵ of letter pairs as shown in Figure 3 where pairs with identical letters are printed as a single letter and pairs of corresponding non-identical letters are separated by a colon.

⁴ Many finite-state tools interpret the digit 0 as a null string or epsilon. Often all traces of such a null string are lost in finite-state operations. In the two-level framework, this is not desired, and it is safer to delete the zero symbols Ø explicitly when desired.

⁵ The conversion was done simply by:

```
hfst-strings2fst -i new-old-words.text |
hfst-fst2strings -X print-space -X print-pairs -o new-old-pairs.text
```

```

k:c a u p u n k:g i ∅:h i n
k:c a u p u n k:g i i:j n
k:c o r v:w e s:∅ s a:∅
k:c u k:c k o ∅:i
k:c u o l e m a a n
k:c u o l e m a:∅ a n
k:c u o l e m a n
k:c u o l l e:u i t t e n
k:c u u l k:c o:∅ o:a n
k:c u u l i t t e:a
k ä r s i m ä n
k ä r s i m ä:∅ ä n

```

Figure 3: Some example words with zeros added and aligned letter by letter and shown as a sequence of letter pairs

Once we have the aligned pairs, we compute a list of different pairs and their frequencies as in Figure 4. The pairs end up as the declaration of the alphabet in the two-level rule grammar. The frequencies guide the authoring of rules and can be directly used for weighting alternative analyses.

158 a	2 e:ö	5 j:∅	130 n	84 s	60 u	5 ö
39 a:∅	22 e:∅	23 k	61 o	2 s:n	4 u:∅	1 ∅:d
1 b	2 f:p	53 k:c	2 o:a	16 s:z	1 v	1 ∅:e
9 d	3 g	12 k:g	7 o:∅	16 s:∅	1 v:f	3 ∅:g
97 e	26 h	6 k:x	30 p	102 t	2 v:g	4 ∅:h
3 e:a	122 i	54 l	5 p:b	29 t:d	34 v:w	1 ∅:i
3 e:i	4 i:j	1 l:∅	1 p:w	1 t:l	17 y	1 ∅:n
1 e:u	12 i:∅	36 m	1 p:∅	1 t:r	1 y:∅	1 ∅:s
1 e:ä	6 j	3 m:∅	30 r	1 t:∅	44 ä	4 ∅:t
					22 ä:∅	

Figure 4: Frequencies of the letter pairs found in the aligned example words

5. Automatic alignment

One may add further examples at the later stages of the research. One may also want to remove some examples, if they turn out not to represent any general patterns. To facilitate the maintenance of the collection of examples, an automatic character by character alignment was constructed, see also Koskenniemi (2017). Such an automatic procedure for character by character alignment is expected to be useful for other purposes, as

well, including computational historical linguistics where it can be used in relating cognate words, cf. Koskeniemi (2013a).

The International Phonetic Alphabet (IPA) presents a general taxonomy for vowels and another for consonants, both based on the articulatory features of sounds. This taxonomy and the features can be utilised in computing approximate distances between sounds. Alphabetic scripts of MSF and OLF can be characterised quite well using the articulatory features of the IPA. For our purposes, only a subset of all features permitted by the IPA is needed.

A short Python script (see Appendix 4) was written for building a weighted finite-state transducer (WFST) out of the IPA features for the letters. For two-valued features, and for the tongue height of vowels and for the place of articulation of consonants, an ad hoc numeric value was assigned to each position.⁶ The distances were computed by adding the absolute values of the differences in each feature. Insertions and deletions of letters were all given a constant fairly long distance. In addition to these systematically computed distances, some individual distances were set. These were needed e.g., in order to guarantee a unique treatment of the shortening of double vowels or consonants. Otherwise one could delete either of the two, and there would be no difference in the overall sum of distances. Thus, a few extra items were added in the distance calculation so that it is always the latter letter of the two that is deleted if any (e.g., a a:∅ rather than a:∅ a). Ambiguities caused by the orthographic conventions, e.g., between k:x s:∅ and k:∅ s:x and gemination (adding a second identical consonant after instead of before the existing one) were resolved in a similar manner.

A Python script was written, see Figure 5 (overleaf), to implement the actual alignment. The script uses the WFST for distances that was created as discussed above. The script reads an example word pair (*w1*, *w2*) at a time, converts the MSF word *w1* into a FST and inserts zero symbols ∅ freely to it. The same is done for the OLF word *w2*. Then, *w1* is composed with the alignment WFST `align` and *w2*: `w1/∅ .o. align .o. w2/∅`.

Out of the many possible string pairs that the resulting WFST represents, only the one with the smallest weight is taken and printed. When testing the alignment procedure, one can assess the relative success of each aligned pair of words. Each pair of words gets a score as the sum of all

⁶ The actual process of aligning appears not to be sensitive to the choice of the distances among vowels and among consonants as long as consonants and vowels are not allowed to correspond to each other (with the exception of semivowels). It would worth while to find well motivated distance measures, maybe using data from historical linguistics.

```

import sys, io, fileinput
import libhfst
tok = libhfst.HfstTokenizer()
algfile = libhfst.HfstInputStream("chardist.fst")
align = algfile.read()
for line in sys.stdin:
    (f1,f2) = line.strip().split(sep=":")
    w1 = libhfst.fst(f1).insert_freely(("0", "0")).minimize()
    w2 = libhfst.fst(f2).insert_freely(("0", "0")).minimize()
    w1.compose(align).compose(w2)
    res = w1.n_best(1).minimize()
    paths = res.extract_paths(output='text')
    print(paths.strip())

```

Figure 5: Python script for aligning words letter by letter

character pair correspondence weights. Very high total weights indicate untypical pairs of characters which may sometimes be an error in the example word pair.

All finite-state functions that were needed for the script were available in the HFST-Python integration. This particular operation appears to be clumsy to perform using the standalone programs or XFST or Foma.⁷

6. Writing the two-level rules

For a more detailed description of two-level rules see Beesley & Karttunen (2003) and Karttunen et al. (1987). For the method of finding contexts for rules, see Koskenniemi (2013b). The rules to be written in this project have a common alphabet which consists of the letter pairs shown above in Figure 4. We have to write a two-level rule for each non-identical pair (unless there is just one alternative, or if we let all alternatives be allowed anywhere). The rules may be written in any order one finds convenient. Let us start with the pair e:a. Gnu Emacs was used for editing of test examples, rules and all other files. The Emacs command `0ccurs` was thus available and used for extracting the right kind of information from the examples in letter pair format as in Figure 6.

These OLF word forms sound like some dialectal forms found even today. It was deduced that the correspondence e:a was restricted to two

⁷ One can do it using the HFST command line programs by converting first the MSF words into a sequence of FSTs by `hfst-strings2fst`, the same for OLF words, and then composing the sequences element by element with the alignment FST.


```

3 matches for "e:a" in buffer: new-old-pairs.text
  71: k:c u u l i t t e:a
 141: t u l e t t e:a
 142: t u l i m m e:a

```

Figure 6: Occurrences of *e:a* in the examples

personal plural endings in verbs. Any other MSF word forms ending in *e* do not have OLF forms with *a* instead. Any letters *e* inside the MSF words are likewise unaffected by this alternation. This rule has no access to the grammatical features, it relies on patterns consisting of letters. Thus, the following rule in Figure 7 was written.

```

"e:a" e:a => [t t | m m] _ .#. ;
!                               k:c u u l i t t e:a
!                               t u l i m m e:a

```

Figure 7: Two-level rule which restricts the positions where MSF *e* may correspond to *a* in OLF

By convention, the examples based on which the rule was designed, were always included as comments to the rule. According to the conventions of the two-level rules, see Karttunen et al. (1987), this rule says that the pair *e:a* may occur only if preceded by *tt* or *mm* and is at the end of a word. Only the context restriction (\Rightarrow) is used, not the double arrow⁸, because there are some words where the stem ends similarly, e.g., *lumme* or *amme* where the final vowel does not change. Even the best and most obvious rules are bound to be ambiguous as long as one only has the surface representations available without any morphological or grammatical knowledge.

One can test the first rule right away after it has been written, as will be explained in the next section. Experienced two-level grammar writers often design a few rules before they test them. So, let us study another letter pair *s:∅* before we proceed to testing, see Figure 8 (overleaf).

It is easy to see two patterns here. A double *ss* in MSF is reduced to a single *s* in OLF and *ks* in MSF words is represented as *x∅* in OLF. Thus, we need a rule with two context parts as in Figure 9 (overleaf).

Each rule is then compiled into a FST using the two-level compiler *hfst-twolc*. All rules together form the two-level grammar which

⁸ A double arrow \Leftrightarrow rule would require that the change is obligatory in the given context.

```

16 matches for "s:Ø" in buffer: new-old-pairs.text
14: e d e s s:Ø ä
25: h a a:Ø h d e s s:Ø a:Ø
26: h a a k:x s:Ø i
31: h e n g e s s:Ø ä:Ø
37: h y v:w ä k:x s:Ø i
52: k:c a n s s:Ø a n s a:Ø
60: k:c o r v:w e s s:Ø a:Ø
80: m u r h e e:Ø l l i s e k:x s:Ø i
83: n i i:j s s:Ø ä
119: s e a s s:Ø a:Ø
126: s y d ä m e s s:Ø ä n s ä:Ø
127: s y n a g o g a s s:Ø a:Ø
129: t a p p a a:Ø k:x s:Ø e n s a:Ø
150: u n e s s:Ø a:Ø
160: v:w a p a a:Ø k:x s:Ø i
172: y k:x s:Ø i n ä n s ä:Ø
    
```

Figure 8: Occurrences of s:Ø in the examples

```

"s:Ø" s:Ø => s _ ;
!           e d e s s:Ø ä
           s e a s s:Ø a:Ø
           :x _ ;
!           h a a k:x s:Ø i
    
```

Figure 9: Two-level rule for restricting the deletion of s

is compiled into a sequence of such rule transducers. If one has forgotten or mixed some punctuation in the rules when writing the grammar, there will be error messages with a pointer to the probable location and cause of the error. The grammar writer is expected to correct the error and recompile.

7. Validating the rules against examples

There is a facility for testing two-level grammars. There is a special program, `hfst-pair-test` which checks whether the grammar accepts all examples given as sequences of letter pairs. The same `Makefile` which compiles the rules, performs this check right away. The program reports

any inconsistencies, e.g., character pairs occurring in other contexts than those allowed by the rules or misaligned words resulting in character pairs not allowed by the grammar.

Two familiar concepts from information retrieval are used here with a specific interpretation. *Recall* means here the proportion of OLF words that will get the correct MSF word among the results of the analysis (no matter how many wrong alternatives were produced). *Precision* means here the proportion of correct MSF results among all proposed results for a set of OLF words, e.g., all word tokens in the corpus. Recall and precision can equally well be used for the inverse relation, i.e., from the modern words to the old words.

One ought to remember that the testing of pair string examples only detects problems where the rules are too restrictive. Initially, before we have any rules, all examples would pass the check. Using just a few rules, one could retrieve all old forms for a modern form (as long as they participate in those alternations that were present in the examples). But the initial grammar has a very poor precision. A modern word corresponds to very many (possibly infinitely many) old words and vice versa. As we write more rules in our two-level grammar, the recall can only decrease, but every new rule improves the precision.

If one finds new types of regularities during the process, one ought to add new word pairs to the examples. New letter pairs can then be introduced, aligned and tested in the examples.

8. Standalone testing of the grammar

When one has rules for all letter pairs, the two-level grammar can be tested in a new manner. One can now generate tentative OLF forms from the MSF ones. One gets several results per each modern word. Using unweighted rules, all results of such generation are equal. There is one trivial weighting that can be used here to prioritise resulting words more likely: to use the statistics we have from the example words as in Figure 4. A short Python script is used for computing a WFST from the frequencies. Intersecting the weighted transducer with intersected rule FST gives us a new rule WFST. This one can be safely tested by inputting MSF words to it and selecting at most N , say 20, best results. If the correct one is among the top results, the rules seem to do the right thing. See the transcript in Figure 10 (overleaf) where one can see what the grammar generates out of a few modern words.

The weighted rule transducer can be inverted and thereafter tested in the same way. In the present project, the mapping from OLF to MSF

```

$ hfst-strings2fst | hfst-compose -2 intro.fst | \
  hfst-compose -2 new2old-one-w.fst | \
  hfst-compose -2 delete.fst | hfst-project -p output | \
  hfst-fst2strings -w -N 20

>>sija
sija      1.86035
sia       2.12402 +
>>sokeat
sokeat    3.84277
sokiat    8.85742 +
>>ruoskitte
ruoskitte 4.18848 +
ruoskitt  6.3291
ruoskitta 9.20312 +
ruoskite  10.8613
ruoskit   13.002

```

Figure 10: Testing how the plain rules generate tentative OLF word forms out of MSF word forms. The MSF word form as input is marked with >> and the correct results are marked with a plus sign (+).

words is expected to be more ambiguous than the other direction. Thus, the weighting is useful in checking the production of candidate modern forms. Figure 11 shows the 20 first results generated from an old word *isäm* ‘our father’ out of the total of 32 results.

For some other OLF words, there will be many more results, e.g., for *cullainen* ‘golden’, more than 300 results were produced. Even as such, the mapping might be useful in indexing or searching a corpus. One may easily produce a transducer *oldwords* which accepts exactly the word forms in the corpus. Composing the mapping *new2old* used in Figure 10 with *oldwords* could be quite useful. One could build a search facility on this basis which would use modern word forms as search keys and expand it according to *new2old* and do the actual search using the existing OLF words the mapping gives.

It would be impractical to use the above method in existing concordance programs, as it would require the inclusion of all alternatives, even the nonsense modern “word forms” in the index. However, nothing would prevent us from using *new2old* in a front end processor to traditional concordance programs such as Korp, described in e.g., Borin et al. (2012).

```

$ hfst-strings2fst | hfst-compose -2 intro.fst | \
  hfst-compose -2 old2new-one-w.fst | \
  hfst-compose -2 delete.fst | hfst-project -p output | \
  hfst-fst2strings -w -N 20

>>isäm
isäm      1.23633
isääm     2.94141
isäme     3.82129
issäm     4.31348
iisäm     4.84863
isääme    5.52637
issääm    6.01855
iisääm    6.55371
issäme    6.89844
isämme    7.40625 +
iisäme    7.43359
iissäm    7.92578
issääme   8.60352
isääme    9.11133 +
iisääme   9.13867
iissääm   9.63086
issämme   10.4834
iissäme   10.5107
iisämme   11.0186
issääme   12.1885

```

Figure 11: A test where we see the first 20 results that the inverted rules generate out of one OLF word *isäm*. The correct results are marked with a plus (+).

9. Combining the grammar with OMORFI

As we noticed above, the rules are quite ambiguous when generating tentative modern word forms from an OLF word form. We have a lot of candidates, among which the correct one is hidden. Most of the noise words are non-words in MSF. Thus, it is a natural idea to filter the noisy output of the rules using a spell-checker for MSF.

OMORFI is a finite-state morphological analyser which is open source and freely available, cf. Pirinen (2015). It uses the same HFST tools, so it was easy to combine it with other transducers used in this study. For further information on the HFST morphological tools, see e.g., Lindén et al. (2011). OMORFI is distributed both as source code and as binary

FSTs.⁹ The source form consists of more than 300 files and appears fairly complicated. More than a dozen Makefiles are needed for building the FST that recognises Finnish word forms. Therefore, it was easier to use the binary transducer which comes with the package even if there would have been an obvious need to modify the lexicon and rules to better suit the needs of this project.

The transducer `finnish-analyze.fst` takes a Finnish word form as its input and outputs its analyses as a combination of a base form and the morphosyntactic features characterising the grammatical form, e.g., as in Figure 12.

```
$ hfst-strings2fst \<
  hfst-compose -2 finnish-analysis.fst \<
  hfst-fst2strings
  >>kuutamoilta
  kuutamoilta:kuutamo N Abl Pl
  kuutamoilta:kuutamo#ilta N Nom Sg
```

Figure 12: Morphological analysis using plain OMORFI. Two outputs are generated from the input *kuutamoilta* which is either ‘from moonlights’ or ‘moonlight’ + ‘evening’. Note the word boundary in the second result.

The morphosyntactic features are not needed for the filtering of the noise words from the set of candidates that the rule transducer generates. Only the input side of the transducer is needed for the selection of acceptable word forms of MSF. One can simply drop the output part and keep the input side of the analysis FST.¹⁰

The mapping all the way from OLF word forms into valid MSF word forms is the composition of four transducers in a sequence, see Figure 13. One may run these as a run-time pipeline using separate HFST programs or one may compose them in advance for efficiency.

The combination of the steps in Figure 13 does roughly what was expected. If we feed the OLF words in Figure 1 to it, each old word will be expanded to several possible MSF word candidates, and the analyser will then filter out all but those candidates that it considers acceptable MSF word forms, as is seen in Figure 14.

⁹ The FSTs distributed were in a so called fast lookup form. One can convert them back to the HFST standard form and then modify and manipulate them for the needs of this project.

¹⁰ Projection is made using the command `hfst-project -p input`.

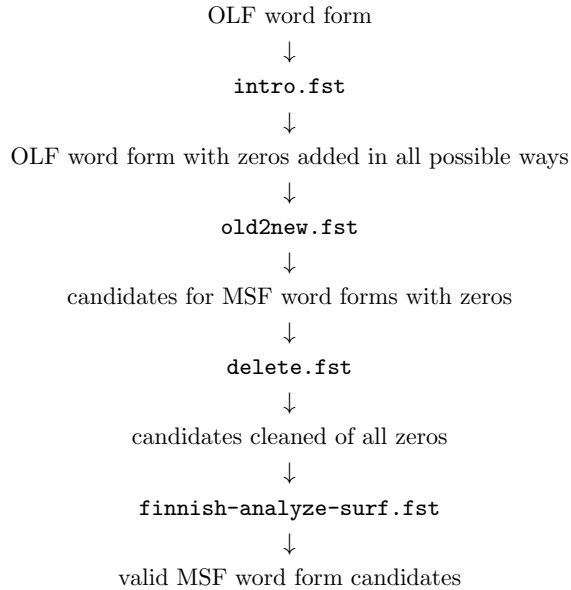


Figure 13: Producing MSF word form candidates out of OLF word forms

caupungihin [kaupunkihiin, kaupunkiin]
 caupungijn [kaupunkiin, kaupunkiini]
 corwes [korvessa]
 cuckoi [kukko]
 cuolemaan [kuolemaan, kuolemaani]
 cuoleman [kuolemaan, kuolemaani, kuoleman, kuolemana,
 kuolemani, kuolleemman]
 cuolluitten [kuolleitten, kuolleitteni]
 cuulcan [kuulkoon]
 cuulitta [kuulitta, kuulitte]
 kärsimän [kärsimän, kärsimäni, kärsimään, kärsimääni]

Figure 14: Analyses of some example words using the two-level grammar and filtering with OMORFI

It is to be noted that most of the modern word forms offered by the sequence are quite acceptable. In particular, all correct interpretations that we wanted are present. In addition to the desired results, there are some artificial words. One of them is the very first result *kaupunkihiin* ‘to the city’ which looks odd. It turns out to be a compound of *kaupunki* ‘town’ and *hiki* ‘sweat’ which is a nonsense word. Another extra result is

kuulitta ‘you heard’, is also an odd compound of kuu ‘moon’ and litta (a children’s play e.g., with a ball).

The number of compound boundaries in a word form would be useful as a criterion for excluding less likely analyses. Unfortunately, when using OMORFI, this information is only available when one reduces the word forms all the way to their base forms. With some Python scripting and processing, the knowledge about the number of compound boundaries can be used at the right place. One first produces a list of all pairs where the first component is the OLF word form and the second component is the analyses OMORFI accepts from the many candidates that the rules propose. The following pairs are in the long list:

```
aitais:aitaiisi
aitais:aitaisi
aitais:aitasi
aitais:aittäisi
aitais:aittäisi
aitais:aittäisi
```

The next step is to analyse the right parts again which were once already accepted by OMORFI, and we get a list containing entries like the following:

```
aitaiisi:aita#iisi A Pos Nom Sg
aitaisi:aidata V Cond Act ConNeg
aitaisi:aidata V Cond Act Sg3
aitaisi:aita#isi N Nom Sg
aitasi:aidata V Pst Act Sg3
aitasi:aita N Gen Sg PxSg2
aitasi:aita N Nom Pl PxSg2
aitasi:aita N Nom Sg PxSg2
aittäisi:aittä#iisi A Pos Nom Sg
aittäisi:aittä#isi N Nom Sg
aittäisi:aittä N Gen Pl PxSg2
```

From these pairs, we only use the number of word boundaries # in the stem that is on the right. For each surface form we store the least number of boundaries its base form analyses have. The first one has only a compound analysis, so it gets the count 1. The next, *aitaisi* ‘of your barn(s)’ or ‘of your fence(s)’ has three analyses, two without boundaries and one with one boundary, so it gets the count 0:


```
aitaiisi 1
aitaisi 0
aitasi 0
aittaiisi 1
aittaiisi 0
aittasi 0
```

Now one can return to the processing of the result pairs where the left part is the OLF word and the right part is a word form proposed by the rules and accepted by OMORFI. For each OLF word, we now have a list of candidate MSF words. We can fairly safely drop some candidate MSF word forms by using their compound boundary count as computed above. We throw away all candidates which have more compound boundaries than the one that has the least number of them. Thus, we start from the following list of modern forms for the OLF word form *aitais*:

```
aitais 1 [aitaiisi, aitaisi, aitasi, aittaiisi, aittaiisi, aittasi]
```

According to the counts we computed, the first and the fourth have a boundary count 1 and the rest has no boundaries. Thus, we drop the first and the fourth, and get the final result which now contains only acceptable words and no artificial constructions:

```
aitais 1 [aitaisi, aitasi, aittaiisi, aittasi]
```

This processing sounds complicated,¹¹ but it is motivated by the fact that OMORFI produces a lot of extra analyses using its liberal compounding mechanism. Anyway, the Python script which does the trick, is short, fast and straightforward.

10. Reducing to the base forms

Normally, OMORFI reduces word forms to their base forms, and base forms would be often even better for searching and indexing than the word forms themselves. Thus, in parallel to the operations in the previous section, the candidate MSF word forms were filtered and reduced to their base forms. This list had the same kinds of problems with the liberal compounding of OMORFI as we saw in the previous section. The artificial

¹¹ It would also be possible to handle the stem counts using WFSTs. One may convert the list of MSF forms and compound part counts into a weighted transducer which accepts the modern forms and use the weight as a criterion for exclusion.

compounds could be removed in the same way, in fact more easily as the compound boundaries were present in the base forms directly. Before the filtering, the results for a base form *alendamisest* ‘from lowering’ looked like the following:

```
alen#da#miss#eesti  'sale'+ 'da'+ 'miss'+ 'Estonian'
alen#da#miss#este  'sale'+ 'da'+ 'miss'+ 'obstacle'
alenta  'to lower'
alentaminen  'lowering'
alentamis#eesti  'lowering'+ 'Estonian'
alentamis#este  'lowering'+ 'obstacle'
alen#tamminen  'sale'+ 'made of oak'
alen#tammis#eesti  'sale'+ 'made of oak'+ 'Estonian'
alen#tammis#este  'sale'+ 'made of oak'+ 'obstacle'
```

Again, the filtering program considers this set of candidate MSF base forms. It finds two candidates with no compound boundaries and seven with one or two boundaries. The program throws away those seven and keeps the two. So the result for *alendamisest* becomes:

```
alenta
alentaminen
```

11. Tuning the two-level rules

At this point the rules have been tested against the examples and they have been used separately for some manually typed words in order to assess the precision of the rules, i.e., how many unwanted analyses they produce. We have tools for reducing OLF word forms into MSF word forms and also to MSF base forms. Now one can see what the rules and OMORFI together actually do to the masses of words of the *Biblia 1642* corpus.

One can expect that some rules are too permissive. This will show up as too many candidate MSF words. On the other hand, some rules might have too narrow context conditions, which will be seen as some OLF words left without the desired candidate words. It is also possible that some regular phenomena were not present in the example words. Then we have no applicable rules and many OLF words remain without the desired candidate MSF words. In the two first cases, we must consider revising the two-level rules we have written. In the last case, we must select further example word pairs and write yet another rule and test it.

In order to check what actually exists in the the *Biblia 1642* corpus, three files were used: the source text itself, an alphabetical list of distinct OLF word forms in the corpus, and a list of reversed OLF word forms

(sorted starting from the last character). Using the Gnu/Linux `less` and `egrep` commands, one got quick answers to questions such as: “Are there many other words similar to this one?” or “Is this really a form of the word I think?”.

The tuning consumed more time than the writing of the initial two-level grammar. It was also more demanding because one must check that changes in rules do not have negative effects, such as dropping some desired candidate words which were previously correctly generated. For this purpose, the changes in the rules were always checked by producing a separate new list and comparing it against a previous full list of analyses¹². If the differences were all for the better, then the new rules were accepted, and the new lists taken as the new benchmark for the following changes. Some of the new or lost analyses required checking from the corpus or the lists of old words as mentioned above.

The tuning required partial knowledge of the language in the corpus and was made by Kimmo Koskenniemi. An overall sense of present day Finnish and some familiarity with Finnish dialects seemed to be sufficient for finding generalisations and adequate context characterisations. Just one OLF word form (*käätyxi*, ‘that has been turned’ could not be interpreted by looking at the *Biblia 1642* occurrences. One had to look it up in a more recent Bible translation.

All changes of the rules were automatically checked against the collection of hand-selected example word pairs. Any discrepancies were immediately detected and the rule violating some word pair was identified. After correcting the rule that failed, the rules were recompiled, retested and the full lists were recomputed. A handful of new example words were included in this process. The original and the new examples were used to test the rules thereafter at every cycle.

There appears to be no clear limit how long one can tune a grammar. After a certain level is reached the return of each cycle diminishes. Many of the remaining shortcomings could be better solved if one could have a different morphological analyser for Finnish. In particular, one would like to modify the compounding mechanism, make the derivational capacity more productive, and use a morphophonemic representation for MSF as the basis for rules. Then one would have access to many relevant conditions for determining the forms of the OLF.¹³ Such a re-implementation of Finnish

¹² The checking was done using the Gnu/Linux `comm` program.

¹³ OMORFI is open source and readily available for modifications. It was designed, however, without morphophonemes or explicit indication for alternating phonemes. OMORFI is very good for e.g., spell checking and even for generating inflected forms

morphological analysis would be motivated also when applying two-level methods to historical linguistics of Finno-Ugric languages, see Koskenniemi (2013a).

A couple of cases occurred where a new letter pair and an entirely new rule had to be established. That posed no major problem as long as the main principles of letter alignment and correspondences remained unchanged. With a few new example word pairs, there were no particular problems.

A common question that arose was to decide whether the rejection of MSF word forms was an error or a feature of OMORFI. The analyser is committed to obey the guidelines for word inflection as described in *Kielitoimiston sanakirja*¹⁴ (2006) which is also available as a net service.¹⁵ In most cases, a fifty years earlier norm of MSF would have better suited the needs of this project.

12. Evaluation of the mapping

The rules were developed using a set of example words. So the discussion of the success and the shortcomings of the mappings cannot be estimated by testing with the same words. One can assess how the mapping covers the vocabulary of the corpus by taking a sample of the list of all distinct word forms in the corpus, i.e., some 26,500 words. This list consist mostly of infrequent words. Half of them are hapax legomena, i.e., occurring only once. Less than 5,000 words occur more than five times in the corpus. Two 100 word samples were selected, one out of the full list of distinct word forms and another from a list consisting of word forms occurring at least six times in the corpus. Both samples were made out of the respective total list by first skipping some entries and then proceeding with even intervals (the length of the list divided by 100). A third sample was made from the running text.

12.1. Proper nouns and abbreviations

The Biblia 1642 corpus contains plenty of proper names, biblical and other. Names of persons and places occur typically fairly few times and only

e.g., in machine translation. Modifying it for the present purposes would be as difficult as building a new analyser.

¹⁴ The dictionary of the Institute for the Languages of Finland.

¹⁵ <http://www.kielitoimistonsanakirja.fi/>

within a short passage of text. There are two kinds of problems concerning them. Dictionaries lack most of them, so the filtering could not work properly. Most proper names are unlike normal Finnish words, and the orthography used in writing them differed from that of normal OLF words. Proper names are often written as in Swedish or German and not adjusted to Finnish.

By mistake, some material, such as references to other parts of the Bible remained in the corpus although the intention was to exclude them all. This happened probably because such markings had more variable forms than was expected. The abbreviations included in this way are not valid OLF words and not a target of this study.

Thus, the proper nouns and abbreviations do occur in the samples, but could be ignored in the results. Proper nouns, and many other words were written with capital letters in *Biblia 1642*. In addition, capital letters are found in the corpus in unusual positions, e.g., both as the first and the second letters. Precise normalisation of the corpus was not a goal of this project, so nothing was done beyond forcing all text to lower case.

12.2. Words occurring more than five times

The result of testing a sample of 100 word forms from the list of word forms occurring at least six times in the corpus is given in Appendix 1. The following is a summary of the results with this sample:

- Eight biblical proper names or abbreviations were missed.¹⁶
- In addition, six OLF word forms were left without a proper analysis: an obsolete form *käätyxi* of the verb *käännetyksi* ‘turned’, ‘converted’; slightly archaic inflections *löytty* ‘found’, *saatit* ‘you might’, ‘you escorted’, *sijpein* ‘of wings’, *vartioidzit* ‘they/you guarded’; a word form *mixette* ‘why not (you)’ accidentally missing from OMORFI. One may assume that these and similar words would be correctly analysed, if the filtering morphological analyser could be modified so that it accepts older inflectional forms.

¹⁶ Four of the eight would be accepted by OMORFI if given with a capital letter: proper names *Amoksen*, *Saul*, *Jerusalemista*, and the roman numeral *XI*. The proper name *Gedalia* would not have been in OMORFI, neither the two abbreviations *Cap* and *Ioh* accidentally included in the list.

12.3. All word forms in the corpus

The full list of the other 100 word sample which was taken from the total list of all word forms occurring in the corpus is given in Appendix 2. The following is a summary of the results:

- There were 11 proper names or abbreviations which were rejected: *Ahabin*, *Bath*, *Giledassa*, *Ismaelille*, *Kanaaneri*, *Kyrenestä*, *Magnus*, *Moph*, *Pet* (= *Petrus*), *Pilatus*, *Publiuksella*. See the discussion above.
- In addition, 16 words did not get the proper analysis, 10 of them occurred just once in the corpus. Two fairly frequent words (*ettäs*, ‘you not’, *synteins*, ‘of their sins’) were not accepted by OMORFI and therefore were lost, and so was *poismenit* ‘you went away’ which is nowadays written as two words.
- Out of the remaining 13 unanalysed words, one *onnettomudexen* ‘to his misfortune’ was analysed to the correct base form but not to the less frequent form actually used in the corpus.
- For three words, one could search further for a possible revision of the rules: *cauhiuttain*, *julgista* ‘priech’, not ‘public’, *tervehdimmä* ‘we welcome(ed)’.
- Many missed OLF words might be better handled by revising the lexicon of the morphological analyser rather than the rules developed for this paper. There is no guarantee that rule revisions would really improve the recall. The OLF word *onnettomudexen* = *onnettomuudeksenne* ‘to your misfortune’ is a case in the point. It would be tempting to include a rule which would allow the recognition of the plural second person possessive suffix by deleting the two final letters. One ought to be careful, though: the rule would over-generate because there are many other words ending similarly and not so many occurrences of this suffix.¹⁷ One would need access to the morphophonological level of MSF in order to describe this suffix accurately.
- Altogether some 165 analyses were produced for the 100 word forms. About 79 of them were exact matches to the actual word in the

¹⁷ It is possible to modify the endings in OMORFI although one has to make changes in many places of the lexicon.

corpus. Ten proposed analyses were unacceptable (six as artificial compounds, four as guesses for proper names).

12.4. Sample of word tokens from the running text

The two above tests estimated how well the method covers the vocabulary. Another aspect is, how well the method covers the text, i.e., how large a portion of word tokens in the running text would get a proper base form by which the place could be retrieved. For this purpose, a sample of 100 words was made, starting with a small offset and stepping through the text at equal intervals. A summary of results with this test:

- Four word tokens were proper names: *Babelin*, *Efraimin*, *Israelin*, *Maria*; two were abbreviations: *Reg*, *XXI* and were left without a correct analysis.
- Three tokens were left without a proper analysis: *iohtunut* ‘caused by’, *murehitit* ‘you worried’, *sijhenasti* ‘till then’. The *j:i* is a very rare correspondence, and the last two ones are ungrammatical in MSF.
- The remaining 91 words of this sample were given, among others, the correct analysis.

One may speculate that frequent words are more common in samples of running text than in samples from lists of distinct word forms. Therefore, it would be expected that the rules and OMORFI perform better with such samples.

13. Conclusion

On the whole, the authors consider the precision and recall of the combination of the two-level rules and OMORFI successful, somewhat better than was expected. Spending more time with the rules and tuning the context conditions would not have a significant effect on the performance. By making the conditions looser, one may improve the recall at the expense of precision. With some manually compiled lists and by paying attention to the capital letters, one could handle the proper names much better.

colmas 26 +kolmas
 costaman 10 koostamaan, koostamaani, koostaman, koostamani,
 +kostamaan, kostamaani, +kostaman, kostamani
 cuitengin 328 +kuitenkin, kuittenkin
 cunnias 9 kunniaasi, +kunniasi, +kunniaassa
 cuulemma 7 kuulemma, +kuulemme, kuulleemme, kuullemme
 duomidzeman 20 +tuomitsemaan, tuomitsemaani, +tuomitseman,
 tuomitsemani
 egyptiläisten 10 +egyptiläisten, egyptiläisteni
 engelille 8 +enkelille
 epäjumalain 22 +epäjumalain, epäjumalaini
 että 2808 +että
 § gedalia 7 =Gedalia
 harwat 15 +harvat
 heitän 9 +heitän, =heidät, heittäne, *heittäni, *heittään,
 *heitäni, *heitään
 hetke 14 +hetkeä
 huones 35 +huoneesi, +huoneessa
 hywä 160 +hyvä, +hyvää, =hyvät
 hävitetän 14 hävitettäne, hävitettäni, hävitettään,
 +hävitetään, hävitteettäni, hävitteettään
 ihmisest 12 +ihmisestä
 § ioh 21 =abbreviation (not part of text)
 itkemän 17 +itkemän, itkemäni, +itkemään, itkemääni
 § jerusalemist 58 =Jerusalemista
 jolle 14 +jolle
 judalaisista 12 +juutalaisista
 jumalinen 9 +jumalinen, jumalineen, jumalineni
 kedolla 53 +kedolla, ketolla
 kircasta 7 kirkasta, +kirkastaa
 kylään 7 +kylään, kylääni, kyyllään, kyylään, kyylääni
 käsiwartens 14 käsivarteensa, +käsivartensa, käsivarttensa
 @ käätyxi 21 =käätyksi (obsolete inflection pro 'käännetyksi')
 laskeman 7 +laskeman, +laskemaan, laskemaani, laskemani
 lewitat 11 +leviitat
 luetan 14 luetan, +luetaan, luettane
 lyödyxi 6 +lyödyksi
 @ löytty 13 =löytty (obsolete inflection pro 'löydetty')
 mailma 57 +maailma, +maailmaa
 menewät 44 +menevät
 miehens 11 mieheensä, +miehensä
 @ mixette 6 =miksette (OK in MSF but OMORFI rejects)
 muu 30 +muu
 neljäkymmendä 16 +neljäkymmentä, neljääkymmentä
 nimittä 12 nimittä, +nimittää
 nähä 88 +nähdä
 oikein 87 +oikein, oikeine, oikeini
 oma 34 +oma, +omaa
 opetuslastens 40 +opetuslastensa
 oxa 11 +oksa, +oksa

paimen 19 +paimen
 palvelusta 22 palvellusta, +palvelusta
 parempi 11 +parempi
 perkelestä 6 +perkeleestä
 pidetän 14 pidettäne, pidettäni, pidettään, +pidetään,
 piteettäni, piteettään
 pohjaisest 8 +pohjaisesta, pohjaisesti
 prophetalle 26 +profeetalle
 puolelle 19 +puolelle
 päiviä 9 +päiviä
 päät 7 +päät, pääte
 rangaisewa 18 +rankaiseva, rankaisevaa
 ristiinnaulidzit 6 ristiinnaulitsit, ristiinnaulitsitte,
 +ristiinnaulitsivat
 ruumis 10 +ruumis, +ruumiisi, ruumiissa, ruumissa
 @ saatit 14 =saatit (old inflection pro 'saattoivat'), *saatit
 sanani 45 +sanani, +sanaani
 § saul 8 =Saul
 seuracunda 38 +seurakunta, +seurakuntaa
 @ sijpein 10 =siipein (old inflection pro 'siipien')
 sisäldä 11 +sisältä, sisältää
 sucucunda 25 +sukukunta, +sukukuntaa, *suukukunta, *suukukuntaa
 suus 9 +suusi, +suussa
 synnyttä 13 synnyttä, +synnyttää
 tahto 243 tahto, +tahtoa, +tahtoo
 tapahtunut 65 +tapahtunut
 tehkät 47 +tehkää
 tie 15 +tie
 toiseens 6 +toiseensa
 tulella 35 +tulella, tulleella, tuulella, tuulleella
 turmelit 6 +turmelit, turmelitte, +turmelivat
 tyttäres 10 tyttäreesi, +tyttäresi, tyttäressä
 täytti 11 +täytti
 uscollinen 10 +uskollinen
 waeli 26 +vaelsi
 waldacundain 15 +valtakuntain, valtakuntaini
 vanhast 8 +vanhasta, vanhasti
 @ wartioidzit 8 =vartioitsivat (old inflection pro 'vartioivat')
 wertauxen 34 vertaukseen, vertaukseeni, +vertauksen, vertaukseni
 vihollisen 15 viholliseen, viholliseeni, +vihollisen, +viholliseni
 woi 223 +voi
 wuorten 22 +vuorten, vuorteni
 § xi 13 =XI (not part of the text)
 yljän 13 +yljän
 ystävä 6 +ystävä, ystävää
 änellä 27 +äänellä

Appendix 2: Sample of all OLF word forms

This sample was taken from the full list of all distinct OLF word forms of the corpus, i.e., each word form appeared only once in the list no matter how many times it occurred in the corpus. The sample starts with the 86th word and proceeds with steps of equal length in the alphabetical list. The markings for proper names or abbreviations (§), words with no analysis (@), manually added interpretations (=), correct analyses (+) and completely irrelevant candidates for MSF words (*) follow the same principles as in the sample in Appendix 1.

§ ahabin 3 =Ahabin
 @ alaidzen 1 =alitse
 andimexi 1 +antimeksi
 arpoja 1 +arpoja
 asuwat 59 +asuvat
 § bath 3 =Bath
 cahleis 5 +kahleissa
 @ cananeri 1 =kanaaneri=Kaanaan asukas, *kanan-erie
 carsi 2 kaarsi, +karsi, karsii
 @ cauhiuttan 1 =kauheuttaan
 @ cherubim 5 =kerubim=kerubi
 colminaisuuden 2 +kolminaisuuden, kolminaisuuteen,
 kolminaisuuteeni, kolminaisuutena, kolminaisuuteni
 cotcatkin 1 kotkaatkin, +kotkatkin
 cullastans 1 +kullastansa, kultastansa
 cuolettaman 1 +kuolettamaan, kuolettamaani, kuolettaman,
 kuolettamana, kuolettamani
 cuurnidzet 2 +kuurnitset, kuurnitsette
 edestäm 8 +edestämme
 eläväin 1 +eläväin, eläväini
 epäjumalista 2 +epäjumalista
 @ ettäs 100 =ettäs (OMORFI)
 § gileadis 5 =Gileadissa
 halkeisit 1 halkeisit, +halkeisivat, halkeisitte
 hedelmälisest 2 +hedelmällisesti, hedelmällisestä
 @ heräjä 1 =heräjä=herää, herääjä, herääjää
 hopiaksi 1 +hopeaksi
 hurscana 1 +hurskaana
 häpiäs 13 +häpeäsi, +häpeässä, +häpeäsi
 ihmisildä 17 +ihmisiltä
 § ismaelille 1 =Ismaelille
 jalca 3 +jalka, +jalkaa
 johdatan 3 +johdatan, johdattane
 @ julgista 1 =julkistaa
 jutteli 7 +jutteli
 kelwatcon 1 +kelvatkoon
 kijtoswirren 4 +kiitos-virren
 kitans 1 kitaansa, +kitansa
 § kyrenist 1 =Kyrenestä
 kätensä 1 +kätensä, +käteensä, kättensä

lainaxi 1 +lainaksi
 laulun 3 +laulun, lauluna, lauluni, lauluun, lauluuni
 lewollisest 1 +levollisesta, levollisesti
 lohduuxellans 1 +lohduuxellansa
 luotat 5 +luotat, luotaat, luotaatte, luotatte
 lähikyläins 1 +lähikyläinsä
 § magnus 3 =Magnus=Suuri, *maa-gnuusi
 @ medzäficunapuulle 1 =metsä-viikuna-puulle
 miehest 4 +miehestä
 § moph 1 =Moph
 muucalaisilda 1 +muucalaisilta
 @ nautitcat 1 =nautitkaa=nauttikaa
 nimiä 1 +nimiä, nimeä, nimeää
 nurisewat 1 +nurisevat, nurisewat
 @ ohrapion 1 =ohrapivon=ohrakourallisen, *ohrapioni
 @ onnettomudexen 1 =onnettomuudeksenne, onnettomuudekseen,
 onnettomuudekseni
 ota 102 +ota, oitta, otaa, +ottaa
 pahenetta 1 +pahenette
 paljastawat 1 +paljastavat
 paransin 2 +paransin
 peljännet 2 peljännet, +peljänneet, peljännette
 § pet 6 =Pet=Petrus=abbreviation, *peet
 § pilatus 58 =Pilatus, *pilattusi, *pilattuusi, *pilatussa
 @ poismenit 1 =poismenit=pois menit
 § publiuxella 1 =Publiuksella
 purpuraan 1 purpuraan, +purppuraan, purppuraani, purpuraani
 päällimmäistä 1 +päällimmäistä
 racastawanans 1 +rakastawanansa
 rascautta 2 +raskautta, +raskauttaa
 riemuhuudon 1 +riemuhuudon, riemuhuutona, riemuhuutoni,
 riemuhuutoon, riemuhuutooni
 rucouxens 3 rukoukseensa, +rukouksensa
 saamme 8 +saamme
 saitte 4 +saitte
 @ saphir 3 =safiiri
 selitetyt 1 +selitetyt
 siellä 1 +siellä
 @ sisälmäisin 1 =sisälmäisiin=sisimmäisiin?
 sotawäke 4 +sotaväkeä
 suremmaxi 1 +suuremmaksi
 @ syndeins 30 =synteinsä/syntiensä (OMORFI)
 syöksemän 1 +syöksemän, syöksemäni, +syöksemään,
 syöksemääni
 taitons 1 +taitonsa, taitoonsa, taittonsa, taittoonsa
 taudist 3 +taudista, tautiista, tautiistä, tautista
 @ terwehdimmä 2 =tervehdimme
 todistaja 4 +todistaja, todistajaa
 tottunet 3 +tottunet, tottuneet, tottunette
 tunnustin 2 +tunnustin

tyhmäin 1 +tyhmäin, tyhmäini
 töillääs 1 +töillääsi, töiltäsi
 uscalda 4 +uskaltaa
 wacudes 1 vakuudessa, vakuuteesi, +vakuutesi
 waiwan 12 +vaivaan, vaivaani, +vaivan, vaivana,
 +vaivani
 wallidzewat 7 +vallitsevat
 warcaudella 1 +varkaudella
 weidzet 1 +veitset
 wiälliset 2 +viälliset
 wihollisillans 1 +vihollisillansa
 wircaan 6 +virkaan, virkaani
 wuoria 7 +vuoria
 wääristä 4 +vääristä, vääristää
 @ ylöllist 1 =ylöllistä=ilkeätä?
 yxi 334 yksi

Appendix 3: Two-level rules

Alphabet

a a:∅ b d e e:a e:i e:u e:ä e:ö e:∅ f:p g h i i:j i:∅
 j j:i j:∅ k k:c k:g k:x l l:∅ m m:∅ n o o:a o:∅ p p:b p:w
 r s s:n s:z s:∅ t t:d t:l t:n t:r t:t:∅ u u:∅ v v:f v:g v:w v:∅
 y y:∅ ä ä:∅ ö ö:ä ö:∅ ∅:d ∅:e ∅:g ∅:h ∅:i ∅:n ∅:s ∅:t ;

Sets

Vowel = a e i o u y ä ö ;
 Cons = b c d f g h j k l m n p r s t v w x z ;

Definitions

Suf1 = [n i: | n s a: | s i: | m m: e:] ;
 Suf2 = (k i n | k a :∅ n) ;
 aSuff = ((a:) (n | Suf1) | i n | i Suf1 | l [l e|t a] (Suf1) |
 n | n a (Suf1) | s [s:|t] a: (Suf1) | t |
 [t:|:t] a (Suf1) | k:x s: e Suf1 | k:x s:∅ i) Suf2 .#. ;
 oSuff = [k:x s: i | l l [a|e (e: n)] (Suf1) | n [a|e] (Suf1) |
 s [s:|t] a: (Suf1) | t | [t:|:t] a (Suf1) |
 t t e n | t t e Suf1] Suf2 .#. ;

Rules

"a:∅" a:∅ => a: _ ;
 ! p a l a j a a:∅
 ! r a a:∅ m a t t u
 :Cons e _ .#. ;
 ! k:c a i k:c k e a:∅
 :Cons o a:∅ _ .#. ;
 ! h o l h o a:∅ a:∅
 :Cons o _ [(a:∅) .#. | :i :s | j [a|i] | :m |

```

      :n .#. | t (t e:) .#. | :w | :Ø* :x] ;
!      k i r o a:Ø i s i t
!      p u t o a:Ø v:w a t
!      p u t o a:Ø m i s i l l a
!      v a i n o a:Ø a:Ø
!      v:w a i n o a:Ø j a n i
!      v:w a i n o a:Ø t t e:a
!      [n | s (s:Ø) | s t] _ .#. ;
!      [s (s:Ø) | s t] _ .#. ;
!      s e a s s:Ø a:Ø
!      a i k:c a n a:Ø
!      e v a n k:g e l i u m i s t a:Ø
!      i v:Ø _ t .#. ;
!      a n t:n o i v:Ø a:Ø t
!      s o t:d i:e i v:Ø a:Ø t

"e:Ø" e:Ø => e _ ;
!      i h m e e:Ø t
!      _ .#. ;
!      i s ä m m:Ø e:Ø
!      _ i t [t e n | a | ä] .#. ;
!      a p o s t o l e:Ø i t t e n
!      i s _ n [a | ä] .#. ;
!      t o i s e:Ø n a
!      a t _ r i [a | o] ;
!      a t e:Ø r i o i t:d s:z i

"e:a" e:a => [t t | m m] _ .#. ;
!      k:c u u l i t t e:a
!      t u l i m m e:a

"e:u" e:u => l _ i t ;
!      k:c u o l l e:u i t t e n

"e:ä" e:ä => n _ m ;
!      e n e:ä m p:b ä Ø:t ä

"e:i" e:i => t a _ n .#. ;
!      o p e t t a e:i n
!      _ [a | ä] ;
!      r u s k e:i a t

"e:ö" e:ö => .#. y l _ n ;
!      y l e:ö n k:c a t:d s:z o

"f:p" f:p => _ Ø:h ;
!      p r o f:p Ø:h e e:Ø t t:Ø a i n

"i:j" i:j => i _ ;
!      n i i:j s s:Ø ä

```

```

"i:Ø" i:Ø => i _ ;
!           r u u m i i:Ø n
!           o _ t ;
!           o s o i:Ø t t i
!           [n | s | s t] _ .#. ;
!           n i m e s i:Ø
!           k:c o l m a s t i:Ø

"i:Ø .#." i:Ø <= i _ .#. ;

"j:i" j:i => .#. o r _ a ;
!           o r j:i a t

"ij:iØ" j:Ø => [k: | l | s: | t:] i _ [a aSuff | o i oSuff] ;
!           k:c a m a r i p a l v:w e l i j:Ø a Ø:t a
!           k:c a u p i t:d s:z i j:Ø a t
!           k:c u l k i j:Ø o i t a
!           k:c u r k i s t e l i j:Ø a t
!           h a k i j:Ø a t
!           h a l t:d i j:Ø a
!           h a l t:d i j:Ø o i l l e
!           h a l l i t:d s:z i j:Ø a
!           j u o k:x s:Ø i j:Ø a n
!           p a l v:w e l i j:Ø a
!           p a l v:w e l i j:Ø o i t a
!           r a n k:g a i s i j:Ø a
!           v:w a a t i j:Ø a n s a:Ø
!           v:w a l e h t e l i j:Ø a t
!           v:w a r t i j:Ø a
!           s i _ [a oSuff | o oSuff | o i t] ;
!           s i j:Ø a
!           t e k i _ ä ;
!           t e k i j:Ø ä

"k:c" k:c => \:k _ [k | (:Ø) [:a | :o | :u] | Ø:h | l | r] ;
!           j a l k:c a i n s a:Ø

"kk:ck" k:c <= _ k ;

"k:g" k:g => n _ ;
!           e n k:g ä
!           .#. t y _ ö ;
!           t y k:g ö s i:Ø

"k:x" k:x <=> _ s:Ø ;
!           h a a k:x s:Ø i

"ll:lØ" l:Ø => l _ ;
!           e h t o o:Ø l l:Ø i s e n

```

```

"mm:mØ" m:Ø => m _ Vowel: ;
!
!           i s ä m m:Ø e:Ø

!"nn:nØ" n:Ø => n _ e:Ø .#. ;
!
!           k ä s i ä n n:Ø e:Ø
!
!           t e i t ä n n:Ø e:Ø
!
!           a j a t u k:x s:Ø i a n n:Ø e:Ø
!
!           k y m m e n e n n:Ø e n

"o:Ø" o:Ø => o: _ ;
!
!           e h t o o:Ø n a

"o:a" o:a => k: _ o: n .#. ;
!
!           k:c u u l k:c o:a o:Ø n
"~ oo:oa" o:o /<= o:a _ ;

"p:b" p:b => m _ ;
!
!           s u u r e m p:b i
!
!           ?? s a p:b Ø:b a t h:t i
!
!           ?? m u Ø:u l p:b e:ä r i n
!
!           ?? t o p:b i a a:Ø n

"p:w" p:w => _ [u | y| ä] :i (s i: (v:Ø ä:Ø t)) .#. ;
!
!           v:w i i:j p:W y i
!
!           l u o p:w u i
!
!           r e p:w ä i s i:Ø
!
!           l e p:w ä Ø:i s i v:Ø ä:Ø t

"pp:pØ" p:Ø => :Vowel (:m | :l | :r) p _ ;
!
!           k:c u m p p:Ø a n i

"s:Ø" s:Ø => s _ ;
!
!           e d e s s:Ø ä
!
!           s e a s s:Ø a:Ø
!
!           :x _ ;
!
!           h a a k:x s:Ø i

"s:n" s:n => s _ [e | u | y] t .#. ;
!
!           n o u:Ø s s:n u t
!
!           k:c a t k:c a i s s:n e e:Ø t

"s:z" s:z => t: _ ;
!
!           e t:d s:z i
!
!           .#. j o k: a i Ø:d _ e ;
!
!           j o k:c a i Ø:d s:z e l l e

"t:d" t:d => [a|e|i|o|u (u:Ø)|y|ä|ö|h|l|n|t:] _ [a|e|i|o|u|y|ä|ö] ;
!
!           p e l t:d o
!
!           _ s:z ;

```



```

!           p a i t : d s : z i
      .# . _ u o m [ a | i ] ;
!           t : d u o m i o n
"lt:ll" t:l => .# . :Cons* :Vowel (:Vowel) l _ ;
!           k : c u l t : l a i n e n

"t:n" t:n => n _ [[o|u] i (v: a:Ø) (t)] .# . ;
!           a n t : n o i
!           i l m a a : Ø n t : n u i
!           a n t : n o i v : Ø a : Ø t
      n _ [[ö|y] i (v: ä:Ø) (t)] .# . ;
!           s y n t : n y i v : Ø ä : Ø t
      n _ [a|ä|y] i s i (v:Ø [a:Ø|ä:Ø]) t .# . ;
!           a n t : n a i s i v : Ø a : Ø t

"rt:rr" t:r => r _ a i s ;
!           k : c u m a r t : r a i s i : Ø

"t:Ø" t:Ø => t _ ;
!           p r o f : p Ø : h e e : Ø t t : Ø a i n

"u:Ø" u:Ø => u _ ;
!           h a l t : d u u : Ø n
!           p a k : c a n a l l i s u u : Ø d e s t a : Ø
      n o _ s s : n ;
!           n o u : Ø s s : n u t

"v:f" v:f => .# . _ [ a n : g | i : c u n a ] ;
!           v : f a n k : g i n a
!           v : f i k : c u n a

"v:g" v:g => u _ u ;
!           s u v : g u n
!           l u v : g u n
!           r i u v : g u l l a

"v:Ø" v:Ø => i _ [ a : Ø | ä : Ø ] t .# . ;
!           s a i s v : Ø a : Ø t

"y:Ø" y:Ø => y _ ;
!           v : w ä ä r y y : Ø t t ä

"ä:Ø" ä:Ø => ä _ ;
!           k ä ä : Ø r m e e : Ø n
      [ e | n s | s s : Ø | s t ] _ .# . ;
!           h e t k e ä : Ø
!           n ä k ö n s ä : Ø
!           h e n g e s s : Ø ä : Ø
      .# . t i e t _ k [ ä ä : Ø Ø : t | ö ] ;
!           t i e t ä : Ø k ä ä : Ø Ø : t

```

```

          i v:Ø _ t .#. ;
!          k ä ä n t:n i v:Ø ä:Ø t

"ö:Ø" ö:Ø => k ö: _ [n | t] .#. ;
!          ä l k ö ö:Ø n

"ö:ä" ö:ä => _ ö: n .#. ;
!          ä l k ö:ä ö:Ø n

"- öö:äö" ö:ö /<= ö:ä _ ;

"Ø:d" Ø:d => _ s:z ;
!          j o k:c a i Ø:d s:z e l l e
!          j o u O:d s:z e n
!          ä k k:Ø i Ø:d s:z e l t ä

"Ø:e" Ø:e => .#. [e :d :z | k ä r s | k ä ä r | p y :Ø h k |
                  r u o :c k | s a l | s o t: | v: a a t:] _ i .#. ;
!          e t:d s:z Ø:e i
!          k ä r s Ø:e i
!          k ä ä r Ø:e i
!          p y y:Ø h k Ø:e i
!          r u o k:c k Ø:e i
!          s a l l Ø:e i
!          s o t:d Ø:e i
!          s o t Ø:e i
!          v:w a a t:d Ø:e i

"Ø:g" Ø:g => .#. [a i|a l|j a l p a l t e|k:c o (x)|r u o|h u o|n ä]
                _ [a|e|o|u|y|ö] ;
!          a i Ø:g o i t
!          a l Ø:g u s t a
!          h u o Ø:g a t a
!          j a Ø:g a t t e
!          k:c o Ø:g o s s:Ø a:Ø
!          k:c o Ø:g o l l a
!          k:k o r Ø:g o t a n
!          n ä Ø:g y n
!          n ä Ø:g ö n
!          p a Ø:g o s t a:Ø
!          r u o Ø:g o n
!          t e Ø:g o i l l a
!          v:w a a ':g a l l a

"Ø:h" Ø:h => f:p _ ;
!          f:p Ø:h a r i s e u s t e n
          a _ a n .#. ;
!          j u h l a Ø:h a n
          e _ e n .#. ;
!          h ä n e Ø:h e n

```

```

i _ i n .#. ;
!           k:c a r i Ø:h i n
o _ o n .#. ;
!           a r m o Ø:h o n
u _ u n .#. ;
!           l o p p u Ø:h u n
ä _ ä n .#. ;
!           e l ä m ä Ø:h ä n
ö _ ö n .#. ;
!           k i v:w i s t ö Ø:h ö n
.#. k:c _ r i s t ;
!           k:c Ø:h r i s t u s

"asi:ais" Ø:i => [a|ä] _ s i:Ø ;
!           a v:w a Ø:i s i:Ø
!           [a|ä] _ s i v:Ø [a:Ø|ä:Ø] t .#. ;
!           l e p:w ä Ø:i s i v:Ø ä:Ø t
k:c u k:c k o _ .#. ;
!           k:c u k:c k o Ø:i
o _ n u t ;
!           a i k:c o Ø:i n u t

"Ø:n" Ø:n => t:d s:z e _ .#. ;
!           y l i t:d s:z e Ø:n
!           o h i t:d s:z e Ø:n
!           l ä p i t:d s:z e Ø:n
!           e d i t:d s:z e Ø:n
!           a l a i t:d s:z e Ø:n

"Ø:s" Ø:s => .#. :c a n s _ [:a | :o] ;
!           k:c a n s Ø:s a n

"Ø:t" Ø:t => k: [a a:Ø | ä ä:Ø] _ .#. ;
!           a n t:d a k:c a a:Ø Ø:t
!           Vowel: Cons:+ Vowel:+ Cons:+ [a|o|i] _ a .#. ;
!           a s i a Ø:t a
!           p a h e m p:b i Ø:t a
!           Vowel: Cons:+ Vowel:+ Cons:+ [ä|i] _ ä .#. ;
!           k y y n ä r ä Ø:t ä
"Ø:w" Ø:w => l _ o i [l | s] ;
!           j a l Ø:w o i l l a

```

Appendix 4: Distances for automatic character by character alignment

The following short Python program builds a WFST which relates MSF word forms to OLF word forms. The resulting WFST is used in the alignment script in Figure 5. The WFST restricts the character by character matching by rejecting most consonant to vowel and vowel to consonant correspondences. Furthermore it gives penalty weights to letter

correspondences depending on how many of their features differ and how much they differ. The numerical values used in the program are more or less arbitrary and one may tune them in order to improve the accuracy.

The program was made for written Finnish language, but one could modify it in order to use it for some other languages. In particular, it would be interesting to extend it so that it would cover phonetic IPA representations of any language.

```

"""Produces a kind of a distance matrix between
characters in an alphabet."""
import sys, io
import libfst
algfile = libfst.HfstOutputStream(filename="chardist.fst")

vowels = {
    'i':('Close','Front','Unrounded'),
    'y':('Close','Front','Rounded'),
    'u':('Close','Back','Rounded'),
    'e':('Mid','Front','Unrounded'),
    'ö':('Mid','Front','Rounded'),
    'o':('Mid','Back','Rounded'),
    'ä':('Open','Front','Unrounded'),
    'a':('Open','Back','Unrounded')
}

cmo = {'Close':1, 'Mid':2, 'Open':3}
fb = {'Front':1, 'Back':2}
ur = {'Unrounded':1, 'Rounded':2}

consonants = {
    'm':('Bilab','Voiced','Nasal'),
    'p':('Bilab','Unvoiced','Stop'),
    'b':('Bilab','Voiced','Stop'),
    'v':('Labdent','Voiced','Fricative'),
    'w':('Labdent','Voiced','Fricative'),
    'f':('Labdent','Unvoiced','Fricative'),
    'n':('Alveolar','Voiced','Nasal'),
    't':('Alveolar','Unvoiced','Stop'),
    'd':('Alveolar','Voiced','Stop'),
    's':('Alveolar','Unvoiced','Sibilant'),
    'l':('Alveolar','Voiced','Lateral'),
    'r':('Alveolar','Voiced','Tremulant'),
    'j':('Velar','Voiced','Approximant'),
    'k':('Velar','Unvoiced','Stop'),
    'g':('Velar','Voiced','Stop'),
    'h':('Glottal','Unvoiced','Fricative')}
pos = {'Bilab':1, 'Labdent':1, 'Alveolar':2, 'Velar':3, 'Glottal':4}
voic = {'Unvoiced':1, 'Voiced':2}
def cmodist(x1, x2):
    """Computes a distance of Close/Mid/Open and returns it"""
    return abs(cmo[x2] - cmo[x1])

```

```

def posdist(x1, x2):
    """Computes a distance of articulation position and returns it"""
    return abs(pos[x2] - pos[x1])

def adist(x1, x2):
    """Computes a distance between symbols"""
    return (0 if x1 == x2 else 1)

def printlset(lset):
    """Print the set of letters and their features"""
    ll = sorted(lset.keys());
    flist = []
    for l in ll:
        (x,y,z) = lset[l]
        flist.append("{} : {},{},{}".format(l, x, y, z))
    print('\n'.join(flist))

def featmetr(lset1, lset2, f1, f2, f3):
    """Compute all metric distances between letters in d1 and d2
    according to their features."""
    ll1 = sorted(lset1.keys())
    ll2 = sorted(lset2.keys())
    ml = []
    for l1 in ll1:
        (x1,y1,z1) = lset1[l1]
        for l2 in ll2:
            (x2,y2,z2) = lset2[l2]
            dist = f1(x1,x2) + f2(y1,y2) + f3(z1,z2)
            ml.append("{}: {}:{}".format(l1,l2,dist))
    return (ml)

vvlist = featmetr(vowels, vowels, cmodist, adist, adist)
cclist = featmetr(consonants, consonants, posdist, adist, adist)
vowl = sorted(vowels.keys())
cons = sorted(consonants.keys())
letters = sorted(vowl + cons)
dellist = ['{}:0:{}'.format(l,3) for l in letters]
epelist = ['0: {}:{}'.format(l,3) for l in letters]
dbllist = ['{} 0: {}:{}'.format(l,1,2) for l in letters]
sholist = ['{} {} :0:{}'.format(l,1,2) for l in letters]

speclist = ['k:c::0 k::0', 'k:x s:0::0', 't:d s::0', '0:d s::3',
            'i:j::1', 'j:i::1', 'i j:0::0', 'i i:j::0',
            'f:p 0:h::0', 'u:v::1', 'v:u::1', 'u:w::1', 'k:c::1',
            '[o:0 o:?:::5]']
all = vvlist + cclist + dbllist +
      sholist + dellist + epelist + speclist
re = '[{}]*'.format(' | '.join(all))

algfst = libhfst.regex(re)

```

```

algfile.write(algfst)
algfile.flush()
algfile.close()

```

References

- Beesley, K. R. and L. Karttunen. 2003. Two-level rule compiler.
<https://web.stanford.edu/~laurik/.book2software/>
- Biblia. 1642. Biblia. Se on: Coco Pyhä Ramattu, Suomexi. Pämamattuin, Hebreaan ja Grekan jälke: Esipuhetten, Marginaliaain, Concordantiaain, Selitösten ja Registerein cansa. Pipping 42, Henrik Keyser, Stockholmis.
- Borin, L., M. Forsberg and J. Roxendal. 2012. Korp – the corpus infrastructure of Språkbanken. In Proceedings of LREC 2012. Istanbul: ELRA, 474–478.
- Grönros, E.-R. and Kotimaisten kielten tutkimuskeskus. 2006. Kielitoimiston sanakirja. Helsinki: Kotimaisten kielten tutkimuskeskus.
<http://www.kielitoimistonsanakirja.fi/>
- Karttunen, L., K. Koskenniemi and R. M. Kaplan. 1987. A compiler for two-level phonological rules. In M. Dalrymple, R. Kaplan, L. Karttunen, K. Koskenniemi, S. Shaio and M. Wescoat (eds.) Tools for morphological analysis. Palo Alto, CA: Center for the Study of Language and Information, Stanford University. 1–61.
- Koskenniemi, K. 2013a. Finite-state relations between two historically closely related languages. In Proceedings of the Workshop on Computational Historical Linguistics at NODALIDA 2013. Linköping: Linköping University Electronic Press, 53–53.
- Koskenniemi, K. 2013b. An informal discovery procedure for two-level rules. *Journal of Language Modelling* 1. 155–188.
- Koskenniemi, K. 2017. Aligning phonemes using finite-state methods. In Proceedings of the 21st Nordic Conference on Computational Linguistics. Gothenburg: Association for Computational Linguistics, 56–64.
- Lindén, K., E. Axelson, S. Hardwick, T. A. Pirinen and M. Silfverberg. 2011. HFST – Framework for compiling and applying morphologies. In C. Mahlow and M. Piotrowski (eds.) Systems and Frameworks for Computational Morphology 2011 (SFCM-2011). Berlin & New York: Springer. 67–85.
- Pirinen, T. A. 2015. Omorfi – Free and open source morphological lexical database for Finnish. In Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015). Vilnius: Linköping University Electronic Press, 313–315.
- Sadeniemi, M. (ed.). 1951–1961. Nykysuomen sanakirja. Porvoo & Helsinki: WSOY.