Effort versus performance tradeoff in Uralic lemmatisers

Nicholas Howell and **Maria Bibaeva**
National Research University Higher School of Economics, Moscow, Russia

Francis M. Tyers
National Research University Higher School of Economics, Moscow, Russia
Indiana University, Bloomington, IN, United States

Abstract

Lemmatisers in Uralic languages are required for dictionary lookup, an important task for language learners. We explore how to decide which of the rule-based and unsupervised categories is more efficient to invest in. We present a comparison of rule-based and unsupervised lemmatisers, derived from the Giellatekno finite-state morphology project and the Morfessor surface segmenter trained on Wikipedia, respectively. The comparison spanned six Uralic languages, from relatively high-resource (Finnish) to extremely low-resource (Uralic languages of Russia). Performance is measured by dictionary lookup and vocabulary reduction tasks on the Wikipedia corpora. Linguistic input was quantified, for rule-based as quantity of source code and state machine complexity, and for unsupervised as the size of the training corpus; these are normalised against Finnish. Most languages show performance improving with linguistic input. Future work will produce quantitative estimates for the relationship between corpus size, ruleset size, and lemmatisation performance.

Abstract

Uralilaisten kielten sanakirjahakuihin tarvitaan lemmatisoijia, jotka tulevat tarpeeseen kielenoppijan etsiessä sanan perusmuotoa. Tutkimme miten päästään selville sääntöpohjaisten ja ohjastamattomien lemmatisoijien luokittelun tehokkuudesta ja mihin pitää sijoittaa lisää työtä. Esittelemme vertailun Giellateknon äärellistilaisen morfologian projektista derivoiduista sääntöpohjaisista lemmatisoijista sekä Morfessor -pintasegmentoijaprojektin ohjastamattomista lemmatisoijista, jotka on opetettu Wikipedia-aineistoilla. Vertailu koskee kuutta uralilaista kieltä, joista suomi edustaa suhteellisen suuriresurssista kieltä ja Venäjän uralilaiset kielet edustavat erityisen vähäresurssisia kieliä. Suoritusta mitataan sanakirjahakuja sanastovähentämistehtävillä Wikipediakorpuksilla. Kielellistä syötettä kvantifioitiin siten, että lähdekoodi- ja äärellistilakoneen monipuolisuutta pidettiin sääntöpohjaisten lemmatisoijien mittana, ja ohjastamattomien lemmatisoijien mittana pidettiin opetuskorpusta. Näiden molempien mittausta normalisoitiin suomen arvoilla. Valtaosa kielistä näyttää suoriutuvan tehtävästä paranevalla tavalla sitä mukaa, kun kielellistä syötettä lisätään. Tulevassa työssä tehdään kvantitatiivisia arviointeja korpuskoon, säännöstösuuruuden ja lemmatisointisuorituksen välisistä suhteista.

1 Introduction

Lemmatisation is the process of deinflecting a word (the *surface form*) to obtain a normalised, grammatically “neutral” form, called the *lemma*.

A related task is **stemming**, the process of removing affix morphemes from a word, reducing it to the intersection of all surface forms of the same lemma.

These two operations have finer (meaning more informative) variants: morphological analysis (producing the lemma plus list of morphological tags) and surface segmentation (producing the stem plus list of affixes). Still, a given surface form may have several possible analyses and several possible segmentations.

Uralic languages are highly agglutinative, that is, inflection is often performed by appending suffixes to the lemma. For such languages, stemming and lemmatisation agree, allowing one dimension of comparison between morphological analysers and surface segmenters.

Such agglutinative languages typically do not have all surface forms listed in a dictionary; users wishing to look up a word must lemmatise before performing the lookup. Software tools (Johnson et al., 2013) are being developed to combine the lemmatisation and lookup operations.

Further, most Uralic languages are low-resourced, meaning large corpora (necessary for the training of some analysers and segmenters) are not readily available. In such cases, software engineers, linguists and system designers must decide whether to invest effort in obtaining a large enough corpus for statistical methods or in writing rulesets for a rule-based system.

In this article we explore this trade-off, comparing rule-based and statistical stemmers across several Uralic languages (with varying levels of resources), using a number of proxies for "model effort".

For rule-based systems, we evaluate the Giellatekno (Moshagen et al., 2014) finite-state morphological transducers, exploring model effort through ruleset length, and number of states of the transducer.

For statistical systems, we evaluate Morfessor (Virpioja et al., 2013) surface segementer models along with training corpus size.

We hope to provide guidance on the question, "given an agglutinative language with a corpus of N words, how much effort might a rule-based analyser require to be better than a statistical segmenter at lemmatisation?"

1.1 Reading Guide

The most interesting results of this work are the figures shown in Section 5.4, where effort proxies are plotted against several measures of performance (normalised against Finnish). The efficient reader may wish to look at these first, looking up the various quantities afterwards.

For (brief) information on the languages involved, see Section 2; to read about the morphological analysers and statistical segmenters used, see Section 3.

Discussion and advisement on directions for future work conclude the article in Section 6. The entire project is reproducible, and will be made available before publication.

2 Languages

The languages used for the experiments in this paper are all of the Uralic group. These languages are typologically agglutinative with predominantly suffixing morphology. The following paragraphs give a brief introduction to each of the languages.

Finnish (ISO-639-3 `fin`) is the majority and official (together with Swedish) language of Finland. It is in the Finnic group of Uralic languages, and has an estimate of around 6 million speakers worldwide. The language, like other Uralic languages spoken in the more western regions of the language area has predominantly SVO word order and NP-internal agreement.

Komi-Zyrian (ISO-639-3 `kpv`; often simply referred to as Komi) is one of the major varieties of the Komi macrolanguage of the Permic group of Uralic languages. It is spoken by the Komi-Zyrians, the most populous ethnic subgroup of the Komi peoples in the Uralic regions of the Russian Federation. Komi languages are spoken by an estimated $220,00$ people, and are co-official with Russian in the Komi Republic and the Perm Krai territory of the Russian Federation.

Moksha (ISO-639-3 `mdf`) is one of the two Mordvinic languges, the other being Erzya; the two share co-official status with Russian in the Mordovia Republic of the Russian Federation. There are an estimated $2,000$ speakers of Moksha, and it is dominant in the Western part of Mordovia.

Meadow Mari (ISO-639-3 `mhr`, also known as Eastern Mari) is one of the minor languages of Russia belonging to the Finno-Volgaic group of the Uralic family. After Russian, it is the second-most spoken language of the Mari El Republic in the Russian Federation, and an estimated $500,000$ speakers globally. Meadow Mari is co-official with Hill Mari and Russian in the Mari El Republic.

Hill Mari (ISO-639-3 `mrj`; also known as Western Mari) is one of the minor languages of Russia belonging to the Finno-Volgaic group of the Uralic family, with an estimated $30,000$ speakers. It is closely related to Meadow Mari (ISO-639-3 `mhr`, also known as Eastern Mari, and Hill Mari is sometimes regarded as a dialect of Meadow Mari. Both languages are co-official with Russian in the Mari El Republic.

Erzya (ISO-639-3 `myv`) is one of the two Mordvinic languages, the other being Moksha, which are traditionally spoken in scattered villages throughout the Volga Region and former Russian Empire by well over a million in the beginning of the 20th century and down to approximately half a million according to the 2010 census. Together with Moksha and Russian, it shares co-official status in the Mordovia Republic of the Russian Federation.[1]

North Sámi (ISO-639-3 `sme`) belongs to the Samic branch of the Uralic languages. It is spoken in the Northern parts of Norway, Sweden and Finland by approximately 24.700 people, and it has, alongside the national language, some official status in the municipalities and counties where it is spoken. North Sámi speakers are bilingual in their mother tongue and in their respective national language, many also speak the neighbouring official language. It is primarily an SVO language with limited NP-internal agreement. Of all the languages studied it has the most complex phonological processes.

Udmurt (ISO-639-3 `udm`) is a Uralic language in the Permic subgroup spoken in the Volga area of the Russian Federation. It is co-official with Russian in the Republic of Udmurtia. As of 2010 it has around 340,000 native speakers.

Grammatically as with the other languages it is agglutinative, with 15 noun cases, seven of which are locative cases. It has two numbers, singular and plural and a series of possessive suffixes which decline for three persons and two numbers.

In terms of word order typology, the language is SOV, like many of the other Uralic languages of the Russian Federation. There are a number of grammars of the language in Russian and in English, e.g. Winkler (2001).

[1] `https://efo.revues.org/1829`

Table 1: Giellatekno bilingual dictionary sizes, in words.

Language	Lexemes
fin	19012
kpv	43362
mdf	28953
mhr	53134
mrj	6052
myv	15401
sme	17605
udm	19639

3 Lemmatisers

3.1 Giellatekno transducers

Giellatekno is a research group working on language technology for the Sámi languages. It is based in Tromsø, Norway and works primarily on rule-based language technology, particularly finite-state morphological descriptions and constraint grammars. In addition to the Sámi languages, their open-source infrastructure also contains software and data for many other Uralic languages.

In particular, Giellatekno has produced (Moshagen et al., 2014) finite-state transducers for morphological analysis of our chosen Uralic languages; we use these to extract lemmas from surface forms. When multiple lemmatisations are offered, the highest weight one is chosen. Unaccepted words are treated as already-lemmatised.

3.2 Morfessor

Morfessor (Virpioja et al., 2013) is a class of unsupervised and semi-supervised trainable surface segmentation algorithms; it attempts to find a minimal dictionary of morphemes. We use Wikipedia as training data for this model.

4 Evaluation

4.1 Dictionary task

The stemmers are applied to every word in the corpus, and the resulting stem is looked up in a dictionary. This mimics a user attempting to look up a highlighted word in a dictionary.

Bilingual dictionaries are taken from Giellatekno, with definitions in Russian, Finnish, English, or German. (The actual definitions are not used, just the presence of an entry; we take the union over all dictionaries.) Dictionary sizes are shown in Table 1.

As baseline we take the percentage of words in the corpus which are already in the dictionary. Both token and type counts provided.

4.2 Vocabulary reduction

We apply the lemmatisers to each word of the corpus, and measure the reduction in tokens and types. Lower diversity of post-lemmatisation tokens or types demonstrates that the lemmatiser is identifying more words as having the same lemma.

The distinction between token reduction and type reduction corresponds to a notion of ”user experience”: from the perspective of our tasks, correctly lemmatising a more frequent token is more important than a less frequent token.

4.3 Effort

The effort expended in producing a model is a subjective and qualitative measure; we claim only to provide coarse objective and quantitative proxies for this.

In the case of statistical methods, total effort (which would include the effort of developing the algorithm) is not important for our purposes: we are comparing the specialisation of a statistical method to a particular language with the development of a rule-based model. (Indeed, to fairly compare total effort of the technique, a completely different and perhaps more academic question, we would need to include the general development of rule-based methods.) Thus for statistical methods we include only the size of the corpus used to train the system. In our experiments, this corpus is Wikipedia, which we use (for better or worse) as a proxy for general availability of corpora in a given language on the internet.

For rule-based systems, we must find a measure of the effort. In this article our rule-based systems are all finite-state transducers, compiled from rulesets written by linguists. We choose two proxies for invested effort: the lines of code in all rulesets used in compiling the transducer, and the number of states of the transducer.

The former will count complex and simple rules the same, which the latter may provide insight into. Conversely, a highly powerful rule system may create a great number of states while being simple to write; in this case, the ruleset is a better proxy than the number of states.

4.4 Wikipedia

Wikipedia dumps from `20181201` are used as source corpus; the corpus is split into tokens at word boundaries and tokens which are not purely alphabetical are dropped. Corpus size in tokens, post-processing, is shown in Table 2.

Corpora were randomly divided into training (90% of the corpus) and testing subcorpora (10%); Morfessor models are produced with the training subcorpus, and lemmatiser evaluation is only with the test subcorpus.

5 Results

Our study involves treating the Uralic language as an independent variable; the six languages we consider here do not provide for a very large sample. We attempt to mitigate this by using both traditional and robust statistics; potential “outliers” can then be quantitatively identified. Thus for every mean and standard deviation seen, we will also present the *median* and the *median absolute deviation*.

Table 2: Wikipedia corpus size by language, in alphabetic words.

Language	Tokens	Types
fin	897867	276761
mrj	352521	51420
mhr	15159	6468
myv	11177	5107
sme	9442	6552
udm	7503	4308

For reference: suppose that $\{x_i\}_{i=1}^N$ is a finite set of numbers. If $\{y_i\}_{i=1}^N$ is the same collection, but sorted (so that $y_1 \leq y_2 \leq \cdots \leq y_N$), then the median is

$$\mathrm{med}\{x_i\} = \begin{cases} y_{N/2} & N \text{ is even} \\ \mathrm{mean}\{y_{(N\pm 1)/2}\} & N \text{ is odd} \end{cases}$$

and the median absolute deviation (or for brevity, "median deviation") is

$$\mathrm{mad}\{x_i\} = \mathrm{med}\left\{|x_i - \mathrm{med}\, x_i|\right\}.$$

When we quote means, we will write them as $\mu \pm \sigma$ where μ is the mean and σ the standard deviation of the data. Similarly, for medians we will write $m \pm d$ where m is the median and d the median deviation.

Data with potential outliers can be identified by comparing the median/median deviation and the mean/standard deviation: if they are significantly different (for example, the mean is much further than one standard deviation away from the median, or the median deviation is much smaller than the standard deviation), then attention is likely warranted.

5.1 Dictionary lookup

Results of the dictionary lookup are presented in Table 3.

Cursory inspection shows that while the Giellatekno model for Finnish slightly out-performs the Wikipedia Morfessor model, on average Morfessor provides not only the greatest improvement in token lookup performance (average/median improvement of $1.6/1.5$ versus Giellatekno's $1.4/1.3$), but also more consistent (standard/median deviation of $0.3/0.1$ versus $0.4/0.3$).

We see some limitations in the Morfessor model when projecting to type lookup performance: the value of Morfessor on type lookup is essentially random, hurting as often and as much as it helps: mean and median improvement factors are both 1.0. Compare with Giellatekno, where improvement mean and median are at least one deviation above baseline. We suggest this disparity could be due to our Morfessor model over-stemming rare words, and successfully stemming common words.

5.2 Vocabulary reduction

Vocabulary reduction results are presented in Table 4.

Generally, we see that Morfessor is much more aggressively reducing the vocabulary: average Morfessor reduction is 9% versus Giellatekno's 15%; here North Sámi and Finnish again stand out with Morfessor reducing to 7.2% and 6.5% respectively. Compare with Hill Mari, where reduction is to a mere 11%.

While the performance of Giellatekno is much less dramatic, we still notice that North Sámi and Hill Mari are more than a standard deviation, or more than two median deviations, away from the mean performance. Otherwise, the clustering is fairly tight, with all languages besides North Sámi and Hill Mari within one standard deviation and 1.5 median deviations.

The analysis above shows that our data are affected by outlier models; which of the two measures is nominally more representative of the overall performance landscape could be demonstrated through an increase of sample size, i.e., increasing the number of languages surveyed.

5.3 Effort

The effort quantification is presented in Table 5. Transducer source code complexity, measured in number of transducer states per line of source code, is presented in Table 6. Note that comments are included as part of the "source code"; we consider, for example, explanation of how the code works to count as some of the effort behind the development of the transducer.

Some immediate observations: among the Uralic languages studied here, Finnish is high-resource, but not overwhelmingly: North Sámi compares for transducer size (in number of states), at nearly 2.5 times the median. While Meadow Mari actually has a comparable amount of transducer source code (1.8 million lines of code, about 80% the size of the Finnish transducer), its transducer code is extremely low complexity; see Table 6. Finnish Wikipedia is approximately 2.5 times larger than the next largest, Hill Mari, and nearly 7 times larger than the median; under our assumption, this would indicate that Finnish written material is also much more accessible on the internet than our other Uralic languages.

Among Giellatekno models, Hill Mari transducer is uniformly the lowest-resource of the Uralic languages studied, with very few lines of below-average complexity code written; contrast this with the Morfessor models, where Hill Mari has a respectable $350,000$ tokens. The lowest resource Morfessor model is Udmurt, with only $7,000$ tokens; the Udmurt Giellatekno model is also significantly below-average in resources.

While North Sámi has slightly below-median transducer source size, it has extremely high (eight deviations above median) state complexity, with more than one state for every two lines of code.

5.4 Analysis

See Figures 1, 2, and 3 for plots of effort normalised against Finnish versus performance. Plots are colored by language and marked by the effort quantification method. Note that since "lines of code" and "number of states" are two different measures of the same model,

Table 3: Results of the dictionary lookup task for no-op (NOOP), Morfessor (MF), and Giellatekno transducer (GT). A “hit” means a successful dictionary lookup. Percentage hits (tokens or types) is the percentage of tokens or types in the corpus for which the lemmatiser produces a dictionary word. The “no-op” (NOOP) lemmatiser takes the surface form as-is, and is used as baseline; the last two columns are percentage hits normalised by this.

Language	**Lemmatiser**	**Hits** (thous.)		**% Hits**		**Improvement**	
		tokens	**types**	**tokens**	**types**	**tokens**	**types**
`fin`	NOOP	10.2	2.55	11.0	5.0	-	-
`kpv`		0.5	0.14	43.0	22.0	-	-
`mdf`		2.1	0.75	32.0	19.0	-	-
`mhr`		0.6	0.26	39.0	24.0	-	-
`mrj`		5.4	0.76	15.0	6.0	-	-
`myv`		0.4	0.13	38.0	19.0	-	-
`sme`		0.1	0.08	15.0	10.0	-	-
`udm`		0.2	0.14	31.0	22.0	-	-
average	NOOP	2.0 ± 3.0	0.6 ± 0.8	30.0 ± 10.0	16.0 ± 7.0	-	-
median		0.5 ± 0.3	0.2 ± 0.09	32.0 ± 9.0	19.0 ± 4.0	-	-
`fin`	GT	19.1	3.0	21.0	6.0	1.9	1.2
`kpv`		0.5	0.14	45.0	21.0	1.0	0.9
`mdf`		3.9	1.08	61.0	27.0	1.9	1.4
`mhr`		0.6	0.27	42.0	26.0	1.1	1.1
`mrj`		8.3	0.88	23.0	7.0	1.5	1.1
`myv`		0.4	0.17	38.0	24.0	1.0	1.3
`sme`		0.3	0.14	29.0	17.0	2.0	1.7
`udm`		0.3	0.16	35.0	25.0	1.1	1.1
average	GT	4.0 ± 6.0	0.7 ± 0.9	40.0 ± 10.0	19.0 ± 8.0	1.4 ± 0.4	1.2 ± 0.2
median		0.6 ± 0.3	0.22 ± 0.08	36.0 ± 8.0	22.0 ± 4.0	1.3 ± 0.3	1.2 ± 0.1
`fin`	MORF	18.7	2.4	21.0	5.0	1.8	0.9
`kpv`		0.6	0.18	56.0	27.0	1.3	1.2
`mdf`		3.0	0.63	47.0	16.0	1.5	0.8
`mhr`		0.8	0.24	51.0	23.0	1.3	0.9
`mrj`		8.1	0.57	23.0	5.0	1.5	0.7
`myv`		0.6	0.13	49.0	19.0	1.3	1.0
`sme`		0.3	0.11	34.0	14.0	2.3	1.3
`udm`		0.3	0.15	45.0	24.0	1.4	1.1
average	MORF	4.0 ± 6.0	0.6 ± 0.7	40.0 ± 10.0	16.0 ± 8.0	1.6 ± 0.3	1.0 ± 0.2
median		0.7 ± 0.4	0.21 ± 0.09	46.0 ± 7.0	17.0 ± 6.0	1.5 ± 0.1	1.0 ± 0.1

Table 4: Vocabulary reduction results for no-op (NOOP), Morfessor (MF), and Giellatekno (GT) lemmatisers. The final column gives the reduction factor in vocabulary size: reduction of 1 corresponds to no reduction performed, while 0.01 corresponds to a 100-fold reduction in vocabulary (average of 100 types per lemma). Note that there is no constraint that the "lemmas" produced are dictionary words.

Lang.	**Model**	**Lemmas** (k)	**% Red.**
`fin`	NOOP	264.5	-
`kpv`		4.7	-
`mdf`		18.1	-
`mhr`		6.3	-
`mrj`		46.9	-
`myv`		5.0	-
`sme`		6.5	-
`udm`		4.2	-
average	NOOP	45 ± 84	-
median		6.4 ± 2.0	-
`fin`	GT	41.8	15.8
`kpv`		0.6	13.6
`mdf`		2.9	15.8
`mhr`		1.0	16.6
`mrj`		9.7	20.8
`myv`		0.7	13.4
`sme`		0.8	12.1
`udm`		0.6	14.7
average	GT	7.3 ± 13	15.3 ± 2.5
median		0.9 ± 0.3	15.2 ± 1.5
`fin`	MORF	17.1	6.5
`kpv`		0.4	9.1
`mdf`		1.8	9.9
`mhr`		0.6	9.9
`mrj`		5.2	11.1
`myv`		0.4	8.6
`sme`		0.5	7.2
`udm`		0.4	9.9
average	MORF	3.3 ± 5.4	9.0 ± 1.4
median		0.5 ± 0.1	9.5 ± 0.6

Table 5: Effort quantification; last column is normalized by Finnish. The group 'Mloc' refers to millions of lines of code in the Giellatekno transducer source, including `lexc`, `xfst`, regular expression, constrain grammar, and `twol` code. The group 'kst' is the number (in thousands) of states in the Giellatekno transducer, and 'ktok' is the number (in thousands) of tokens in the Morfessor training corpus. The final column normalises against Finnish.

Lang.	**Model**	**Effort**	**Quan.**	**%** `fin`
`fin`	GT	kst	440	100
`kpv`			150	35
`mdf`			60	13
`mhr`			80	17
`mrj`			50	11
`myv`			110	25
`sme`			540	122
`udm`			60	15
avg.	GT	kst	190 ± 180	40 ± 40
med.			90 ± 40	20 ± 9
`fin`	GT	Mloc	2.3	100.0
`kpv`			0.7	30.0
`mdf`			0.9	40.0
`mhr`			1.8	80.0
`mrj`			0.5	20.0
`myv`			1.2	50.0
`sme`			0.9	40.0
`udm`			0.5	20.0
avg.	GT	Mloc	1.1 ± 0.6	50 ± 30
med.			0.9 ± 0.3	40 ± 10
`fin`	MORF	ktok	898.0	100.0
`kpv`			11.0	1.2
`mdf`			64.0	7.1
`mhr`			15.0	1.7
`mrj`			353.0	39.3
`myv`			11.0	1.2
`sme`			9.0	1.1
`udm`			7.0	0.8
avg.	MORF	ktok	171 ± 296	19.1 ± 33.0
med.			13 ± 5	1.5 ± 0.5

Table 6: Transducer source complexity, in number of states per line of transducer source code. The column "LoC (M)" gives the number of lines of source code, in millions, and "States (k)" the size, in thousands of states of the compiled transducer.

Lang.	**LoC** (M)	**States** (k)	**Complex.**
`fin`	2.3	440.0	0.19
`kpv`	0.7	150.0	0.21
`mdf`	0.9	60.0	0.06
`mhr`	1.8	80.0	0.04
`mrj`	0.5	50.0	0.09
`myv`	1.2	110.0	0.09
`sme`	0.9	540.0	0.63
`udm`	0.5	60.0	0.14
avg.	1.1 ± 0.6	200 ± 200	0.2 ± 0.2
med.	0.9 ± 0.3	90 ± 40	0.12 ± 0.06

their performance is the same.

Figure 1 indicates that for the dictionary lookup task by-token, Morfessor with Wikipedia is more effort-efficient (relative to Finnish) for Komi-Zyrian, Udmurt, North Sámi, Erzya, Meadow Mari, and Giellatekno is more effort-efficient for Hill Mari. Remaining is Moksha, for which performance improvement scales with effort independent of model, and Finnish.

Since we normalise effort against Finnish, we can only observe that the Finnish Giellatekno model performs slightly better than the Finnish Wikipedia Morfessor model; efficiency claims cannot be made.

Figure 2 indicates that for the dictionary lookup task by-token, Morfessor with Wikipedia is more effort-efficient (relative to Finnish) for Komi-Zyrian only; Giellatekno remains more effort-efficient for Hill Mari. Meanwhile, Udmurt, North Sámi, Erzya, and Meadow Mari join Moksha in improvement scaling with effort; the spread in slopes (the rate at which performance improves as effort is increased) is, however, quite large.

Figure 3 shows that, as with lookup performance for tokens, Morfessor dominates vocabulary reduction efficiency, with only Hill Mari scaling with relative effort.

6 Conclusion

6.1 Discussion

There are many interesting things to notice in the effort-performance analysis.

Focusing just on the dictionary task, we find that compared against the same technology for Finnish, the Giellatekno North Sámi (`sme`) transducer has very high performance (relatively small ruleset), due to high rule complexity (the number of states is not very low). It is possible that North Sámi is simply easy to lemmatise, as Morfessor seems to do very well with a small corpus.

Hill Mari (`mrj`) shows predictable performance: relative to Finnish, a small increase in resources (going from 20% or 30% of Finnish resources for the Giellatekno transducer to 40% resources for the Wikipedia corpus) gives a modest increase in performance.

Overall, we see that percent improvement in tasks scales with effort (relative to Finnish) in the type-lookup task; in the token-lookup and vocabulary reduction tasks, performance improvement favours Morfessor. (That is, the Morfessor model has a higher improvement-to-resource ratio, with resources relative to Finnish.) This might be explained by the dramatic spread in Wikipedia corpus sizes used in the Morfessor models: median corpus size is $1.5\% \pm 0.5\%$ the size of Finnish. Thus, improvement of 5% of the Morfessor model is increasing the nominal effort (kilotokens) by a factor of four, for the median corpus; compare with Giellatekno, where median model is 20% or 40% the size of the corresponding Finnish model, depending on the metric used. See the following section for potential avenues to control for this.

6.2 Future work

In the dictionary task, hits/words is lower than unique hits/words (see Section 5.1); this indicates that mislemmatised words are more frequent. Since irregular words are typically high-frequency, we might hypothesize that filtering these would close this gap. If not, it might point out areas for improvement in the lemmatisation algorithm.

We would like to also try other methods of lemmatising. One of the problems with the finite-state transducers is that they have limited capacity for lemmatising words which are not found in the lexicon. It is possible to use guesser techniques such as those described in Lindén (2009), but the accuracy is substantially lower than for hand-written entries. We would like to approach the problem as in Silfverberg and Tyers (2018) and train a sequence-to-sequence LSTM to perform lemmatisation using the finite-state transducer to produce forms for the training process.

There are other statistical methods, in particular byte-pair encoding and adaptor grammars (Johnson et al., 2006), which should be added to the comparison, and addition of further languages should be straightforward.

A more refined understanding of the relationship between size of corpus and Morfessor would give a richer dataset; this could be achieved by decimating the Wikipedia corpus. For truly low-resource languages, additional corpora may be necessary.

Similar refinement could be produced for the Giellatekno transducers using their version history: older versions of the transducers have had less work, and presumably have less source code. A dedicated researcher could compare various editions of the same transducer.

Cross-validation (in the case of Morfessor) and using multiple testing subcorpora would give some idea of the confidence of our performance measurements at the language-level.

Another interesting analysis, which we do not have the space to perform here, would be to normalise performance P, along the model axis m, for example for lan-

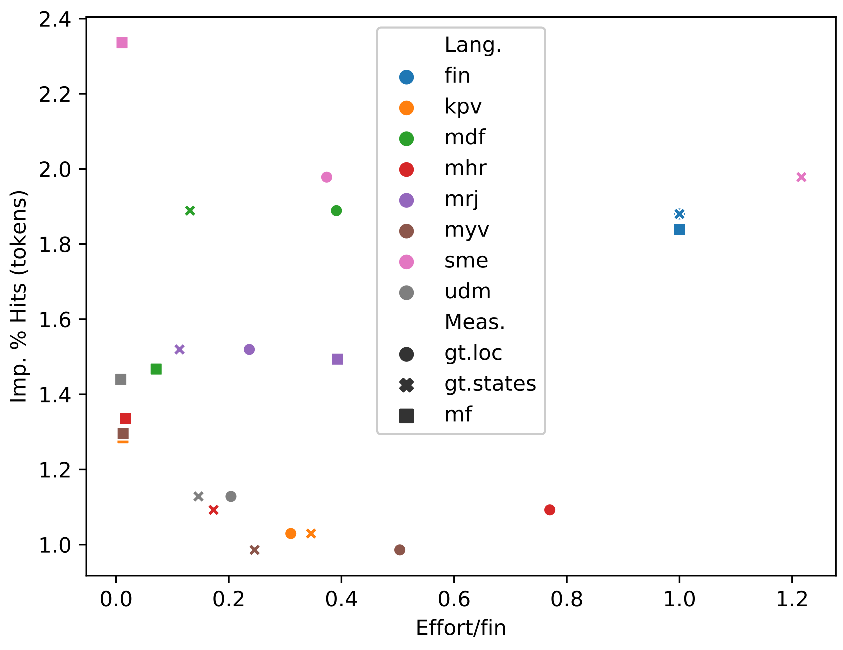

Figure 1: Improvement factor in hit rate in dictionary lookup (by tokens) (see Section 4.1; higher is better) vs. effort relative to Finnish (see Section 4.3; higher is more effort). In general, more effort-efficient models will appear to the upper-left of less effort-efficient models.

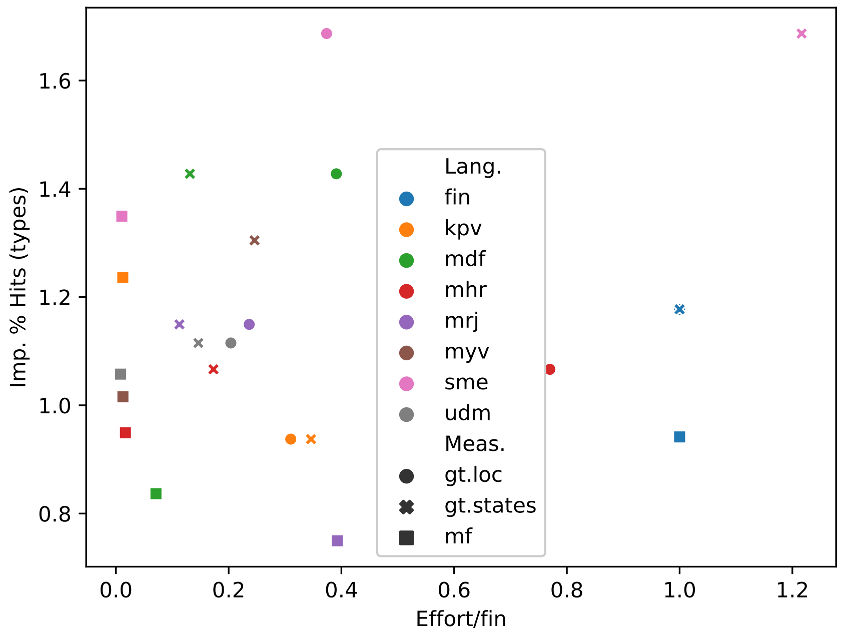

Figure 2: Improvement factor in hit rate in dictionary lookup (by types) (see Section 4.1; higher is better) vs. effort relative to Finnish (see Section 4.3; higher is more effort). In general, more effort-efficient models will appear to the upper-left of less effort-efficient models.

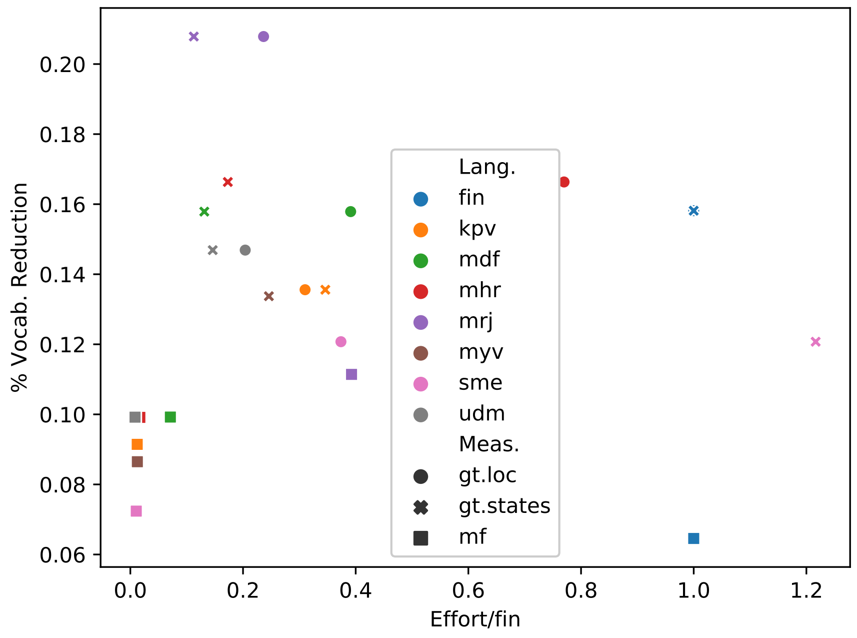

Figure 3: Vocabulary reduction performance in types (see Section 4.2; lower is better) vs. effort relative to Finnish (see Section 4.3; higher is more effort). In general, more effort-efficient models will appear to the lower-left of less effort-efficient models.

guage xxx (normalising against Giellatekno model performance):

$$P^*_{\texttt{xxx},m} = P_{\texttt{xxx},m} \cdot \frac{P_{\texttt{fin},\mathrm{GT}}}{P_{\texttt{fin},m}}$$

This measure, P^*, would always be fixed to 1.0 for Finnish, and would partially control for language-independent performance variation between models. This would then allow study of the distribution over languages of marginal performance improvement with effort.

References

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2006. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Advances in neural information processing systems*, pages 641–648.

Ryan Johnson, Lene Antonsen, and Trond Trosterud. 2013. Using finite state transducers for making efficient reading comprehension dictionaries. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*, 85, pages 59–71.

Krister Lindén. 2009. Guessers for finite-state transducer lexicons. *Computational Linguistics and Intelligent Text Processing 10th International Conference, CICLing 2009*, 5449:158–169.

Sjur Nørstebø Moshagen, Jack Rueter, Tommi Prinen, Trond Trosterud, and Francis Morton Tyers. 2014. Open-source infrastructures for collaborative work on under-resourced languages.

Miikka Silfverberg and Francis M. Tyers. 2018. Data-driven morphological analysis for Uralic languages. In *Proceedings of the 5th International Workshop on Computational Linguistics for the Uralic Languages (IWCLUL 2018)*.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, , and Mikko Kurimo. 2013. Morfessor 2.0: Python Implementation and Extensions for Morfessor Baseline. Technical report, Aalto University.

Eberhard Winkler. 2001. *Udmurt*. Lincom Europa.