Knowledge-light Letter-to-Sound Conversion for Swedish with FST and TBL

Uneson, Marcus

Published in:
Proceedings of Fonetik 2006

2006

Lund University, Centre for Languages & Literature, Dept. of Linguistics & Phonetics 141
Working Papers 52 (2006), 141–144

Knowledge-light Letter-to-Sound Conversion for Swedish with FST and TBL

Marcus Uneson
Dept. of Linguistics and Phonetics, Centre for Languages and Literature, Lund University
marcus.uneson@ling.lu.se

Abstract

This paper describes some exploratory attempts to apply a combination of finite state transducers (FST) and transformation-based learning (TBL, Brill 1992) to the problem of letter-to-sound (LTS) conversion for Swedish. Following Bouma (2000) for Dutch, we employ FST for segmentation of the textual input into groups of letters and a first transcription stage; we feed the output of this step into a TBL system. With this setup, we reach 96.2% correctly transcribed segments with rather restricted means (a small set of hand-crafted rules for the FST stage; a set of 12 templates and a training set of 30kw for the TBL stage).

Observing that quantity is the major error source and that compound morpheme boundaries can be useful for inferring quantity, we exploratively add good precision-low recall compound splitting based on graphotactic constraints. With this simple-minded method, targeting only a subset of the compounds, performance improves to 96.9%.

1 Introduction

A text-to-speech (TTS) system which takes unrestricted text as input will need some strategy for assigning pronunciations to unknown words, typically achieved by a set of letter-to-sound (LTS) rules. Such rules may also help in reducing lexicon size, permitting the deletion of entries whose pronunciation can be correctly predicted from rules alone. Outside the TTS domain, LTS rules may be employed for instance in spelling correction, and automatically induced rules may be interesting for reading research.

Building LTS rules by hand from scratch is easy for some languages (e.g., Finnish, Turkish), but turns out prohibitively laborious in most cases. Data-driven methods include artificial neural networks, decision trees, finite-state methods, hidden Markov models, transformation-based learning and analogy-based reasoning (sometimes in combination). Attempts at fully automatic, data-driven LTS for Swedish include Frid (2003), who reaches 96.9 % correct transcriptions on segment level with a 42000-node decision tree.

2 The present study

The present study tries a knowledge-light approach to LTS conversion, first applied by Bouma (2000) on Dutch, which combines a manually specified segmentation step (by finite-state transducers, FST) and an error-driven machine learning technique (transformation-based learning, TBL). One might think of the first step as redefining the alphabet size, by introducing new, combined letters, and the second as automatic induction of reading rules on that (redefined) alphabet, ordered in sequence of relevance.

For training and evaluation, we used disjoint subsets of a fully morphologically expanded form of Hedelin et al. (1987). The expanded lexicon holds about 770k words (including

proper nouns; these and other words containing characters outside the Swedish alphabet in lowercase were discarded).

2.1 Finite-state transduction (FST)

Many NLP tasks can be cast as string transformation problems, often conveniently attacked with context-sensitive rewrite rules (which can be compiled directly into FST). Here, we first use an FST to segment input into segments or letter groups, rather than individual letters. A segment typically corresponds to a single sound (and may have one member only). Treating a sequence of letters as a group is in principle meaningful whenever doing so leads to more predictable behaviour. Clearly, however, there is an upper limit on the number of groups, if the method should justifiably be called 'knowledge-light'. For Swedish, some segments close at hand are {[s,c,h], [s,s], [s,j], [s,h], [c,k], [k], [k,j]…}; the set used in the experiments described here has about 75 members.

Segmentation is performed on a leftmost, longest basis, i.e., that rule is chosen which results in as early a match as possible, the longest possible one if there are several candidates. All following processing now deals with segments rather than individual letters.

After segmentation, markers for begin- and end-of-word are added, and the (currently around 30) hand-written replace rules are applied, again expressed as transducers or compositions of transducers. These context-sensitive replace rules may encode well-known reading rules (in the case of Swedish, for instance '<k> is pronounced /ɕ/ in front of <e,i,y,ä,ö> morpheme-initially'), or try to capture other partial regularities (Olsson 1998). Most rules deal with vowel quantity and/or the <o> grapheme, reflecting typical difficulties in Swedish orthography. The replacement transducer is implemented such that each segment can be transduced at most once. A set (currently around 60) of context-less, catch-all rules provide default mappings. To illustrate the FST steps, consider the word *skärning* 'cut' after each transduction:

```
input:                  skärning
segment:                sk-ä-r-n-i-ng
marker:                 #-sk-ä-r-n-i-ng-#
transduce:              #-S+<:+r-n-I-N+#
remove marker:          S<:rnIN
```

2.2 Transformation-based learning (TBL)

TBL was first proposed for part-of-speech tagging by Eric Brill (1992). TBL is, generally speaking, a technique for automatic learning of human-readable classification rules. It is especially suited for tasks where the classification of one element depends on properties or features of a small number of other elements in the data, typically the few closest neighbours in a sequence. In contrast to the opaque problem representation in stochastic approaches, such as HMMs, the result of TBL training is a human-readable, ordered list of rules. Application of the rules to new material can again be implemented as FSTs and thus be very fast.

For the present task, we employed the μ-TBL system (Lager 1999). It provides an interface for scripting as well as an interactive environment, and Brill's original algorithm is supplemented by much faster Monte Carlo rule sampling. The templates were taken from Brill (1992), omitting disjunctive contexts (e.g., "A goes to B when C is either 1 or 2 before"), which are less relevant to LTS conversion than to POS tagging.

2.3 Compound segmentation (CS)

The most important error source by far is incorrectly inferred quantity. In contrast to Dutch, for which Bouma reports 99% with the two steps above (and a generally larger setup, with

500 TBL templates), quantity is not explicitly marked in Swedish orthography. One might suspect that this kind of errors might be remedied if compounds and their morpheme boundaries could be identified in a preprocessing step. Many rules are applicable in the beginning or end of morphemes rather than words; we could provide context for more rules if only we knew where the morpheme boundaries are. Compound segmentation (CS) could also help in many difficult cases where the suffix of one component happens to form a letter group when combined with the prefix of the following, as in <matjord>, <polishund>, <bokjägare>. Ideally, segments should not span morpheme boundaries: <sch> should be treated as a segment in <kvälls|schottis> but not in <kvälls|choklad>.

In order to explore this idea while still minimizing dependencies on lexical properties, we implemented a simple compound splitter based on graphotactic constraints. An elaborate variant of such a non-lexicalized method for Swedish was suggested by Brodda (1979). He describes a six-level hierarchy for consonant clusters according to how much information they provide about a possible segmentation point, from certainty (as -rkkl- in <kyrkklocka> 'church bell') to none at all (as -gr- in <vägren> 'verge (road)'). For the purposes of this study, we targeted the safe cases only (on the order of 30-40% of all compounds). Thus, recall is poor but precision good, which at least should be enough to test the hypothesis.

3 Results

3.1 Evaluation measure

The most common LTS evaluation measure is Levenshtein distance between output string and target. For the practical reason of convenient error analysis and comparability with Frid (2003) we follow this, but we note that the measure has severe deficiencies. Thus, all errors are equally important – exchanging [e] for [ə] is considered just as bad as exchanging [t] for [a]. Furthermore, different lexica have different levels of granularity in their transcriptions, leading to rather arbitrary ideas about what 'right' is supposed to mean. For future work, some phonetically motivated distance measure, such as the one suggested by Kondrak (2000), seems a necessary supplement.

Table 1. Results and number of rules for combinations of CS, FST, and TBL. 5-fold cross-validation. Monte Carlo rule sampling. Score threshold (stopping criterion) = 2. The baselines (omitting TBL) are 80.1% (default mappings); 86.6% (FST step only); 88.3% (CS + FST).

Training data		*TBL*		*FST + TBL*		*CS + FST + TBL*	
segments	words	results %	#rules	results %	#rules	results %	#rules
49k	5k	93.8	820	94.9	503	95.5	513
98k	10k	94.1	1131	95.0	761	95.7	809
198k	20k	95.2	1690	95.7	1275	96.5	1250
300k	30k	95.7	2225	96.2	1862	96.9	1756

3.2 Discussion

Some results are given in Table 1. In short, both with and without the TBL steps, adding handwritten rules to the baseline improves system performance (and TBL training time) significantly, as does adding the crude CS algorithm. The number of learnt rules is sometimes high. However, although space constraints do not allow the inclusion of a graph here, rule efficiency declines quickly (as is typical for TBL), and the first few hundred rules are by far the most important. We note that the major error source still is incorrectly inferred quantity.

We have stayed at the segmental level of lexical transcription, with no aim of modelling contextual processes. Although this approach would need (at the very least) postprocessing for many applications, it might be enough for others, such as spelling correction. Result-wise, it

seems that the current approach can challenge Frid's (2003) results (96.9% on a much larger (70kw) training corpus), while still retaining the advantage of the more interpretable rule representation. Frid goes on to predict lexical prosody; we hope to get back to this topic.

4 Future directions

Outside incorporating more sophisticated compound splitting, there are several interesting directions. The template set is currently small. Likewise, the feature set for each corpus position may be extended in other ways, for instance by providing classes of graphemes – C and V is a good place to start, but place or manner of articulation for C and frontness for vowels might also be considered. Such classes might help finding generalizing rules over, say, front vowels or nasals, and might help where data is sparse; the extracted rules are also likely to be more linguistically relevant. If so, segments should preferably be chosen such that they fall clear into classes.

Another, orthogonal approach is "multidimensional" TBL (Florian & Ngai 2001), i.e., TBL with more than one variable. For instance, the establishment of stress pattern may determine phoneme transcription, or the other way round. For most TBL systems, rules can change one, prespecified attribute only (although many attributes may provide context). This is true for µ-TBL as well; however, we are currently considering an extension.

Interesting is also the idea to try to predict quantity and stress reductively, with Constraint Grammar-style reduction rules (i.e., "if Y, remove tag X from the set of possible tags"). Each syllable is assigned an initial set of all possible stress levels, a set which is reduced by positive rules ('ending -<ör># has main stress; thus its predecessor does not') as well as negative ('ending -<lig># never takes stress'). µ-TBL conveniently supports reduction rules.

References

Bouma, G., 2000. A finite state and data oriented method for grapheme to phoneme conversion. *Proceedings of the first conference on North American chapter of the Association for Computational Linguistic*, Seattle, WA.

Brill, E., 1992. A simple rule-based part of speech tagger. *Third Conference on Applied Natural Language Processing,* ACL.

Brodda, B., 1979. Något om de svenska ordens fonotax och morfotax: Iakttagelse med utgångspunkt från automatisk morfologisk analys. *PILUS 38*. Institutionen för lingvistik, Stockholms universitet.

Florian, R. & G. Ngai, 2001. Multidimensional Transformation-Based Learning. *Proceedings of the Fifth Workshop on Computational Language Learning (CoNLL-2001),* Toulouse.

Frid, J., 2003. *Lexical and Acoustic Modelling of Swedish Prosody*. PhD Thesis. Travaux de l'institut de linguistique de Lund 45. Dept. of Linguistics, Lund University.

Hedelin, P., A. Jonsson & P. Lindblad, 1987. *Svenskt uttalslexikon* (3rd ed.). Technical report, Chalmers University of Technology.

Kondrak, G., 2000. A new algorithm for the alignment of phonetic sequences. *Proceedings of the first conference on North American chapter of the ACL*, Morgan Kaufmann Publishers Inc, 288-295.

Lager, T., 1999. The µ-TBL System: Logic Programming Tools for Transformation-Based Learning. *Third International Workshop on Computational Natural Language Learning (CoNLL-1999),* Bergen.

Olsson, L-J., 1998. Specification of phonemic repesentation, Swedish. *DEL 4.1.3 of EC project "SCARRIE Scandinavian proof-reading tools" (LE3-4239).*