Redundant Information Reduction in FST-Based Pronunciation Lexicon Compression

Žiga Golob*, Uliana Dorofeeva*, Jerneja Žganec Gros*, Milena Gros†, Simon Dobrišek†

*Alpineon d.o.o.
Ljubljana, Slovenija
{ziga.golob}@alpineon.si
†Faculty of Electrical Engineering
University of Ljubljana, Slovenia
simon.dobrisek@fe.uni-lj.si

Abstract

Finite-state transducers are frequently used for pronunciation lexicon representations in speech engines, in which memory and processing resources are scarce. This paper proposes a method for further reducing the memory footprint of finite-state transducers representing pronunciation lexicons. A combination of grapheme-to-allophone rules with a finite-state transducer is proposed, which yields a 65% smaller finite-state transducer than conventional approaches.

Zmanjševanje odvečnosti informacije pri kompaktnem zapisu slovarjev izgovorjav s pomočjo končnih pretvornikov

Končni pretvorniki se pogosto uporabljajo za predstavitev slovarjev izgovorjav v sintetizatorjih govora, v katerih je na voljo omejena količina pomnilnika ter procesorske moči. V tem članku je predstavljena metoda za nadaljnjo zmanjševanje potrebne količine pomnilnika za predstavitev slovarja izgovorjav s pomočjo končnega pretvornika. Predlagana je uporaba kombinacije grafemsko-alofonskih pravil ter končnih pretvornikov, kar omogoča izgradnjo 65% manjših končnih pretvornikov kot s pomočjo klasičnih postopkov.

1. Introduction

Consistent and accurate determination of word pronunciation is critical to the success of many speech technology applications. Most state-of-the-art speech engines performing automatic speech recognition (ASR) and text-to-speech synthesis (TTS) rely on lexicons, which contain pronunciation information for many words. To provide maximum coverage of the words, multiword expressions, or even phrases that commonly occur in a given application domain, application-specific words, or phrase pronunciations may be required, especially for application-specific proper nouns such as personal names or names of locations.

Pronunciation lexicons for speech engines contain grapheme and allophone transcription of lexical words (Šef et al., 2004). The "x-sampa-SI-reduced" phonetic alphabet, a subset of the X-SAMPA set as defined for Slovenian (Zemljak et al., 2002), is used in allophone transcriptions. An example of a pronunciation lexicon for a few Slovenian words is shown in Figure 1.

```
...
opera       o:pEra
operah      o:pErah
operam      o:pEram
operama     o:pErama
operami     o:pErami
opere       o:pErE
operi       o:pEri
...
```

Figure 1. An excerpt from a Slovenian pronunciation lexicon.

The storage and run-time processing of pronunciation lexicons is memory consuming, especially for highly inflected languages, where pronunciation lexicons typically contain over one million lexical items. In some systems with limited memory resources—for example, in speech engines for embedded systems or multilingual speech engines—using large pronunciation lexicons is not feasible. In addition, the search time in such lexicons may be long if the lexicons are too large to be stored in the main memory or cache.

Therefore, *memory-efficient representations* of pronunciation lexicons enabling fast lookup are mandatory in order to address these limitations. Another disadvantage of inefficient lexicon representation is the unnecessary use of system resources.

State-of-the-art pronunciation lexicon representation techniques used in speech engines are based on structures called finite-state automata (FSA) as in (Daciuk, 2011) and finite-state transducers (FSTs) (Dobrišek et al., 2010; Rojc et al., 2007); very similar structures are also called tries (Ristov, 2005).

This paper discusses new possibilities for reducing the size of pronunciation lexicon representation using FSTs. We report encouraging results that were obtained by removing redundant information from the allophone transcription prior to building the FST.

2. FST representations of pronunciation lexicons

A FST differs from FSA in that when it accepts a symbol it also outputs another symbol. In this way it can translate an input string into an output string. An example of a FST representing a simple pronunciation lexicon from Table 1 is shown in Figure 2.

GRAPHEMES	ALLOPHONES
hiša	hi:Sa
hišo	hi:SO
hiter	hi:t@r

Table 1. Pronunciation lexicon with three lexical items.

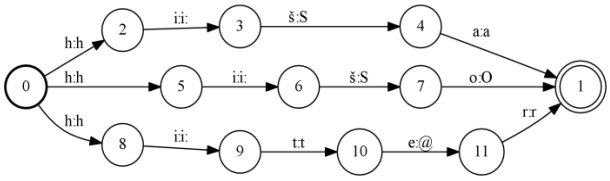

Figure 2. A FST representing a simple pronunciation lexicon from the example in Table 1. By convention, the states are represented as circles and marked with their unique number. The initial state is represented by a bold circle and final states by double circles. An input label i and an output label o are marked on the corresponding directed arc as i : o.

At the beginning, the FST is in its initial state. For every symbol in the input string, the FST changes its state according to the transition function and emits an output symbol. If it moves to a final state when it accepts all the symbols of the input string, we say that it has *accepted* the input string. At that moment the output string, which is composed of all the emitted symbols, becomes valid.

Many efficient algorithms have been developed for FSTs (Cyril et al., 2007), such as union, concatenation, intersection, determinization, minimization, and so on. When using minimization and determinization, the FST becomes a very convenient representation of pronunciation lexicons (Mohri, 1994a; Dobrišek et al., 2009; Rojc et al., 2011). If one excludes all heteronyms (words with the same spelling but different pronunciations), all acyclic FSTs representing pronunciation lexicons can be determinized (Mohri, 1996), and therefore acyclic FSTs are frequently used for pronunciation lexicon representations in speech engines. For representing heteronyms, p-subsequential FSTs can be used (Cyril et al., 2002). For deterministic FSTs, the existence of a minimal FST has been proven(Mohri, 1994b). Hereinafter we denote a minimized and determinized FST as MDFST. Minimization of a FST transforms the FST into an equivalent FST with a minimal number of states. A MDFST also exhibits a minimal number of transitions (Mohri, 1997). The two deterministic FSTs are said to be equivalent if for every sequence of input symbols they generate the same sequence of output symbols.

Table 2 shows the reduction of the number of states by minimization and determinization for the SI-Pron Slovenian pronunciation dictionary containing 1,239,401 lexical items.

TYPE	STATES	TRANSITIONS
FST	11,404,858	12,644,257
MDFST	217,300	517,225

Table 2. Comparison of the number of states and transitions of FST and MDFST representing the SI-Pron pronunciation lexicon.

The FST in Figure 2 shows that there are three possible transitions from the initial state with the same input symbols. In a deterministic FST there is no state with more than one transition with the same input symbol. The advantage of a deterministic FST is lookup speed, which is linearly dependent only on the length of the input string and not dependent on the size of the FST.

In Figure 3 an equivalent FST to the one from Figure 2 is shown, which has been both determinized and minimized (MDFST).

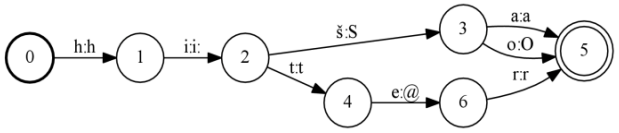

Figure 3. Minimized and determinized FST from Figure 2: MDFST.

3. Language resources

This paper reports results for Slovenian pronunciation lexicon SI-Pron containing 1,239,410 lexical entries (Žganec Gros et al., 2006).

The lexicon contains grapheme and allophone transcriptions for all lexical entries. It also contains additional information such as syllabification, morphological information, and stress positions. Table 3 shows some statistical properties of the lexicon.

LEXICON	WORDS	GRAPHEMES	ALLOPHONES
SI-Pron	1,239k	12,644k	12,713k

Table 3. Statistical properties of the three chosen pronunciation lexicons.

All homographs (words with the same spelling but different meaning) were removed from the lexicon.

4. Experiments and results

It has been reported that for Slovenian it is possible to achieve a grapheme-to-allophone transcription accuracy of 99.1% by using a set of context-dependent rules if the stress position and the transcription of the graphemes *e* and *o* in stressed syllables are known in advance (Gros, 1999).

The idea of the following experiment is to remove all unnecessary information from the pronunciation lexicon and to model the information left with a simpler and possibly lighter FST.

The potential of this approach becomes clearer if one recalls the basic principle of FST minimization. When minimizing a FST, equivalent states are merged. Equivalent states are those that have transitions with the same input and output symbols targeting the same or equivalent states.

According to (Gros, 1999), the necessary information for reconstructing the allophone transcription from the grapheme transcription in Slovenian is the lexical stress position and the transcriptions of the graphemes *e* and *o* in stressed syllables, which we were able to model with only three different symbols in the FST output alphabet. This highly reduced number of different output symbols offers more possibilities for the state transitions to have equal input and output symbols and a greater chance for the possible states to merge.

In this experiment we built a FST representing the pronunciation lexicon emitting information containing only the stress position and the transcription variation of graphemes *e* and *o* in stressed syllables.

In Slovenian, when words of foreign origin are included, the six graphemic vowels *a*, *e*, *i*, *o*, *u*, and *y* along with the reduced vowel schwa @ can be stressed. A lexical item can have multiple lexical stress positions.

The lexical stress position can be modeled implicitly with a FST if, for every accepted grapheme, it emits information on whether it is stressed or unstressed. The reduced vowel @ is not part of the grapheme symbol set. It appears in allophone transcriptions before the consonant *r* when preceded and followed by a consonant (e.g., *potrt* → *pOt@"rt*, *prvi* → *p@"rvi*). Therefore, if the reduced vowel @ is stressed, the FST outputs the information about stress when it accepts the consonant *r*.

The allophone transcriptions of the stressed vowels *e* and *o* can be either close *e:* and *o:* or open *E:* and *O:*. To model this information, for every accepted vowel *e* or *o*, the FST has to emit information on whether the vowel is unstressed, stressed open, or stressed close. Therefore, there are three possible output symbols.

Table 4 shows the possible output symbols of the FST if we group both the stress and the transcription of the graphemes *e* and *o* into one model. Table 5 shows an alternative grouping.

FST INPUT SYMBOL	FST OUTPUT SYMBOL
unstressed grapheme	0
stressed vowel *a, i, u, y*; consonant *r* following a stressed reduced vowel @; stressed vowels *e* and *o* with open transcription	1
stressed vowels *e* and *o* with close transcription	2

Table 4. Modeling the FST information output for stress position and the grapheme e and o transcription. Stressed vowels e and o with open transcriptions are grouped with other stressed vowels.

FST INPUT SYMBOL	FST OUTPUT SYMBOL
unstressed grapheme	0
stressed vowel *a, i, u, y*; consonant *r* following a stressed reduced vowel @; stressed vowels *e* and *o* with close transcription	1
stressed vowels *e* and *o* with open transcription	2

Table 5. Modeling the FST information output for stress position and the grapheme e and o transcription. Stressed vowels e and o with close transcriptions are grouped with other stressed vowels.

Tables 4 and 5 represent two possible mappings between FST input and output symbols. Table 6 shows a few examples of FST input and output strings derived from the mapping presented in Table 4 for a few Slovenian words.

The first two temporary lexicons were constructed for the SI-Pron pronunciation lexicon based on the mappings from Tables 4 and 5. These two lexicons represented the alignments of the symbols (graphemes) in input strings and symbols (numbers) in output strings. Then we built the MDFST from these temporary lexicons using the OpenFST tool.

FST INPUT STRING	FST OUTPUT STRING	ALLOPHONE TRANSCRIPTION
medved	010000	mE:dvEd
prvi	0100	p@"rvi
roža	0200	ro:Za

Table 6. FST output strings for three Slovenian words (*medved, prvi, roža*) and their complete allophone transcription in the last column.

Table 7 shows the comparison between different approaches. The results in Table 7 show a considerable 65% reduction in the number of states (using the mapping from Table 5) compared to MDFST storing the complete allophone transcription.

We also compared the sizes (in MB) of different data structures representing the information stored in pronunciation lexicons. We disregarded the sizes of program code that are necessary to manipulate the data structures because, if implemented properly, they are negligible in size in comparison to the data structures.

ALGORITHM	STATES	TRANSITIONS
FST	11,404,858	12,644,257
MDFST	217,300	517,225
MDFST + information reduction (table 7 mapping)	78,329	246,674
MDFST + information reduction (table 8 mapping)	76,846	242,851
FSA	49,741	155,988

Table 7. Comparison between different approaches to representing the information in the SI-Pron pronunciation lexicon with FST. The finite-state acceptor (FSA) stores the information when the specific input string is valid. It represents the lower bound of the number of states of the FST with the same input alphabet and accepting the same language.

The SI-Pron lexicon as UTF8 encoded text represents the baseline and 100% size. We used the OpenFST tool to build a MDFST representing the complete lexicon information and a MDFST representing only the necessary information for rule-based grapheme-to-allophone conversion.

We also implemented our own more compact representation of the FST similar to the implementation for finite state automata found in (Daciuk, 2011).

It is interesting to compare the lexicon reduction techniques used to standard methods used in text compression, such as zip, even though standard text compression methods are not useful for solving our problem because their data have to be decompressed completely to their full size before they can be used. The results are shown in Table 8.

In its compact form, the MDFST structure representing the reduced information of the SI-Pron pronunciation lexicon is over 40 times smaller than the original UTF8-

encoded text representation as seen in Table 8. It is also three times smaller than the MDFST representing the complete allophone transcription.

	Size [kB]	Size [%]
SI-Pron (UTF8 encoded text)	30,657	100
Compressed SI-Pron (zip)	6,071	19.8
MDFST (OpenFST)	9,948	32.4
MDFST (compact representation)	2,287	7.7
MDFST + information reduction: Table 4 mapping (OpenFST)	4,084	13.3
MDFST + information reduction: Table 4 mapping (compact representation)	708	2.3

Table 8. Size of data structures representing information in the SI-Pron pronunciation lexicon.

5. Conclusion

Finite-state transducers are frequently used for pronunciation lexicon representations in speech engines, in which memory and processing resources are scarce.

A method for further memory footprint reduction of finite-state transducers representing pronunciation lexicons was proposed in the paper. A combination of grapheme-to-allophone rules with a FST yielded a 65% smaller finite-state transducer than conventional approaches.

All the information that can be reconstructed with a set of context-dependent rules was removed from the allophone transcription in the Slovenian pronunciation lexicon. By building the MDFST for the new lexicon, we succeeded in significantly reducing the number of states by 65% and in achieving an implementation over three times smaller.

The proposed method can be used for efficiently representing Slovenian pronunciation lexicon resources; the use of a similar principle could also be considered for other languages with similar pronunciation properties.

6. Acknowledgements

The research work by the first author was partially financed by the European Union, European Social Fund, the framework of the Operational Programme for Human Resources Development for the Period 2007–2013 under contract no. P-MR-10/94.

7. References

Cyril A., Mohri M., 2002. p-Subsequentiable Transducers. Proceedings of the Seventh International Conference on Implementation and Application of Automata (CIAA 2002), Tours, France. pp. 24–34.

Cyril A., Michael R., Johan S., Wojciech S., Mohri M., 2007. OpenFst: A General and Efficient Weighted Finite-State Transducer Library. Proceedings of the 12th International Conference on Implementation and Application of Automata (CIAA 2007). Lecture Notes in Computer Science, Prague, Springer-Verlag, Heidelberg, Germany, 4783: 11-23.

Daciuk J., 2011. Smaller Representation of Finite State Automata. Proceedings of the 16th International Conference on Implementation and Application of Automata, pp. 118–129.

Dobrišek S., Vesnicer B., Mihelič F., 2009. A Sequential Minimization Algorithm for Finite-State Pronunciation Lexicon Models. Proceedings of Interspeech 2009, International Speech Communication Association, Brighton, UK, pp. 720–723.

Dobrišek S., Žibert J., Mihelič F., 2010. Towards the Optimal Minimization of a Pronunciation Dictionary Model. Petr Sojka, Ales Horak, Ivan Kopecek and Karel Pala (Eds.). TSD-2010. Lecture Notes in Computer Science, Brno, Springer, pp. 267–274.

Gros J., Mihelič F., 1999. Acquisition of an Extensive Rule Set for Slovene Grapheme-to-Allophone Transcription. Proceedings 6th European Conference on Speech Communication and Technology. September 5–9. 1999. Eurospeech 1999. Budapest, 5: 2075–2078.

Mohri M., 1994a. Compact Representations by Finite-State Transducers. 32nd Meeting of the Association for Computational Linguistics (ACL '94). Proceedings of the Conference. Las Cruces. NM, pp. 204–209.

Mohri M., 1994b. Minimization of Sequential Transducers. Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching (CPM '94), Maxime Crochemore and Dan Gusfield (Eds.). Vol. 807 of *Lecture Notes in Computer Science*, Asilomar, CA, Springer-Verlag, Berlin, pp. 151–163.

Mohri M., 1996. On Some Applications of Finite-State Automata Theory to Natural Language Processing. *Journal of Natural Language Engineering*, 2: 61–80.

Mohri M., 1997. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 33: 269–311.

Ristov S., 2005. LZ Trie and Dictionary Compression. Jurnual Software-Practice & Experience, pp. 445–465.

Rojc M., Kačič Z., 2007. Time and Space-Efficient Architecture for a Corpus-Based Text-to-Speech Synthesis System. Speech Communication, 49: 230–249.

Rojc M., Mlakar I., 2011. Multilingual and Multimodal Corpus-Based Text-to-Speech System – PLATTOS. Ipšić Ivo (Ed.). Speech and Language Technologies, InTech, Available from: http://www.intechopen.com/books/speech-and-language-technologies/multilingual-and-multimodal-corpus-based-text-to-speech-system-plattos. Accessed 2012 June 6.

Šef T., Gams M., 2004. Data mining for creating accentuation rules, Applied. Artificial Intelligence, vol. 17, pp. 395-410.

Zemljak M., Kačič Z., Dobrišek S, Gros J, Weiss P., 2002, Računalniški simbolni fonetični zapis slovenskega govora. Slavistična revija, 50: 159–169.

Žganec-Gros J., Cvetko-Oresnik V., Jakopin P., 2006. SI-Pron Pronunciation Lexicon: A New Language Resource for Slovenian. Informatica, 30: 447–452.