MODELING PRONUNCIATION VARIATIONS AND COARTICULATION WITH FINITE-STATE TRANSDUCERS IN CSR

Schamai Safra, Gunnar Lehtinen, Karl Huber

Computer Engineering and Networks Laboratory
Swiss Federal Institute of Technology Zurich
Switzerland
{safra,lehtinen,huber}@tik.ee.ethz.ch

ABSTRACT

In this paper we report on how Finite-State Transducers (FST) can be integrated into a CSR system to express the regular context-dependent relationship between a canonical phonemic language model and its possible phonetic realizations, and thus cover many of pronunciation variation and inter- and intra-word coarticulation phenomena. By having a separate intermediate model for those phenomena we keep them out of the higher lexical level and the lower level of acoustic models so each of the levels can be handled separately and can be made more accurate.

We present some experimental results of the use of FSTs in our experimental CSR system ARCOS-G [1]. These FSTs were compiled from combinations of so-called two-level rules which were assembled manually according to a small set of well-known linguistic rules in German.

We then address the question of automatically generating FSTs from examples. Our special focus is to keep the resulting FSTs manageably small by combining the many particular rules learned from examples with few, but more general rules manually collected. Experiments with different approaches are presented.

1. INTRODUCTION

A major problem in continuous speech recognition is a poor language model. Using an inaccurate language model during acoustic model training leads to "broad" acoustic models and thus to lower acoustic discrimination power. In the recognition phase, an inaccurate language model can overrule a correct acoustic decoding.

The particular problem we address in this paper is that the standard phonemic transcription of words of the lexicon does not describe pronunciation variants of words nor intra- and inter-word coarticulation. (To be short, we will name all of these phenomena as "articulation variations" throughout this paper, to indicate any variation from the standard phonemic transcription of words or concatenations of these transcriptions in the case of continuous speech.)

Standard solutions to this problem are either searched for on the lexical level, e.g. using word pronunciation networks instead of a simple phone sequence, or on the acoustic model level, e.g. using "phoneme" models, triphone models etc.

Many of the variation phenomena mentioned are not lexical, though, nor are they adequately addressed by the acoustic level, but should rather be expressed by a regular relationship between an ideal on the phonemic level and its possible realizations on the phonetic level. Here are some examples to illustrate this:

Some articulation variations are indeed lexical in nature in the sense that they only appear in connection with a particular word. For instance, /fYnftsIc/, the German word for "fifty", can be pronounced in some dialects as [fUftsIc], while the same kind of variation hardly ever occurs in other words. Such phenomena are best described on the lexical level, i.e. as alternative transcriptions in a pronunciation lexicon. Other variations, e.g. different dialectal tinges of phones, are best described on the phonetic level.

Many articulation variations, however, do not belong to either category. Rather, they can be best expressed by an intermediate level between the lexical and the acoustic level. This level expresses the regular relationship between a canonical phonemic transcription and every possible phonetic realization thereof. As an example, the word /fʏnftsɪç/ is often pronounced [fʏmftsɪç], neither because this is a variant of this specific word, nor because every /n/ can also be realized as [m], but rather because /n/ in the context ⟨open vowel⟩ _ /f/ is often realized as [m].

For such cases, it seems to be best to express these phenomena explicitly in a separate intermediate model. The introduction of such an intermediate level has several advantages:

- New words can be added by simply indicating their standard phonemic transcription, without the need to explicitly indicate every possible pronunciation variant.
- Word and morpheme boundaries can be marked with special symbols in the phoneme sequence for variations that are restricted to such context (e.g. word-final devoicing in German). However, they can just as well be ignored when those boundaries do not matter. This is a most natural way to express both intra- and inter-word coarticulation when appropriate (e.g. geminate reduction).
- There is no need for the acoustic models to express

This work was partly done within the framework of the COST Action 249

Appeared in *Proceedings of the ESCA Workshop: Modeling Pronunciation Variation for ASR, Rolduc NL, 1998.*

different variants so they can describe narrowly confined phones and thus have better discriminative power.

2. FINITE-STATE TRANSDUCERS

2.1. Rewrite Rules

In order to describe articulation variation, we would like to express something like "n can sometimes be realized as m, but only in some contexts, as ʊ _ f". Rewrite rules seem to offer a formalism to express that kind of statements.

Rewrite-rules in the most general definition—without going into all the formal details—describe which parts of a symbol sequence can be replaced by which others. In order to get a legal output sequence from a given input sequence, any number of a finite set of rewrite rules may be applied any number of times and in any order. The output "can be explained" by the input if a corresponding sequence of rules can be found that transform one into the other. For example, the set of rules

$$\begin{aligned} \text{ndf} &\rightarrow \text{nf} \\ \text{ʊnf} &\rightarrow \text{ʊmf} \\ \text{mf} &\rightarrow \text{mpf} \end{aligned}$$

can explain the above and similar phenomena.

Rewrite rules, however, have some significant drawbacks: Obviously, with rewrite-rules, there is no natural way to separate the input alphabet from the output alphabet, although they can be of a completely different nature; it is a one-way transformation rather than a relation so that there is no simple way to find an inverse rule set nor is there a simple way to find the explaining rule application sequence for any pair of input/output sequences or reject the pair as not having an explanation. And last but not least, it is practically difficult to formulate such a set of rules that would guarantee termination of the search process.

A formalism more appropriate in our view is the so-called Finite-State Transducer (FST). Many ideas involved in using FSTs as a formalism to describe linguistic knowledge go back to work done by Koskenniemi, Kaplan and Karttunen [2, 3], Ritchie et al. [4] and Hoequist and Nolan [5].

2.2. Informal Introduction to FSTs

Finite-State Transducers are well-known computational devices for symbolic processing. Finite-State Transducers (FST) are powerful and computationally efficient yet simple models to express context-dependent regular relationships between two levels of a language. They are used successfully in the symbolic language processing of e.g. translation systems [6] or our TTS System SVOX [7]. In CSR, FSTs are used to combine different levels of the language model by applying FST composition rather than simple replacement, so as to adequately address context-dependency constraints [8].

Without going into technical definitions, FSTs are simply Finite-State automata (FSA) that consume symbol pairs at each transition, where each pair is made from a deep-level symbol and a surface-level symbol, each from a *different* alphabet. If regarded as a recognition device, one can say that in each transition, a surface symbol—an observation unit—is consumed while a deep symbol—a corresponding language symbol[1]—is *emitted*. On both levels, "epsilon" symbols are allowed that symbolize non-emitting respectively non-consuming transitions.

As with simple FSAs, being in a state characterizes the past history—what has been consumed already—and predicts the possible future history. Since there are only a finite number of states, in FSAs that describe infinite languages, at least some states must characterize infinitely many possible "pasts" and "futures". If a FSA should well describe a language, those pasts should be represented by one state which would all predict the same possible futures. In FSTs, being in a state means having emitted one of some particular language symbol sequences and *at the same time* having consumed one of some particular observation unit sequences. Considering the context on both levels allows to make some phenomena dependent on others, but without the need to perform several transformations in a row, as with rewrite-rules.

We found Finite-State Transducers to be an adequate formalism to model the intermediate level described earlier, mainly for these reasons:

- they have enough expressive power to model context-dependent regular relationships between two levels of a language [3], as requested.
- they can quite easily be compiled from a set of two-level rules [3] or two-level regular expressions [9].
- they are straightforward to use and computationally efficient: All well-known properties of and methods for FSAs can be used.
- several FSTs for different subsequent language abstraction levels can be created separately and then merged into one by the FST-composition operation.
- since they define a relation between sequences rather then a transformation from one sequence to another, they are flexible in their application: They can be used in a top-down manner to build a recognition network from phonemic expectations, in a bottom-up manner to suggest phonemic hypotheses from acoustic evidence, or, in a mediating mode, to dynamically match a pair of corresponding sequences from the offered phonemic expectations *and* the acoustic hypotheses found during recognition.

3. THE USE OF FSTS IN ARCOS-G

3.1. The ARCOS-G System

In ARCOS-G ("experimental system for the Automatic Recognition of COntinuous Speech in German"), recognition is done by a combination of knowledge-based and statistical means: on the phonoacoustic level, a statistical decoder delivers a lattice of phonetic hypotheses. On this lattice a parser operates to search for the best-scored global hypothesis fitting the rule-based language model. The parser used

[1] We use the terms "observation unit" and "language symbol" to be illustrative. Of course any two abstraction levels of the language representation could be meant here.

in ARCOS is a chart-parser trimmed to handle uncertain and contradicting hypotheses as delivered by a phonoacoustic decoder. The phonetic decoder used in the experiment described here is a standard HMM recognizer operating in N-best lattice delivery mode.

For representing the morpho-syntactic and lexical knowledge a Definite-Clause Grammar together with a morpheme lexicon is used. The lexicon maps words onto standard phonemic transcriptions, thus the phonemes used are the terminal symbols of the grammar. To mediate between these phonemic symbols and the recognition units delivered by the phonoacoustic decoder, a Finite-State Transducer is used. This FST is actually a composition of three different FSTs each describing a distinct level:

- morphophonemic FST: This FST incorporates the phonemic changes that may or have to take place when composing morpheme to words. Morpheme and word boundaries, flexion types and other relevant context information are encoded into the morphemes with special symbols. These symbols disappear on the phonemic level but enables or forces the transducer to select the appropriate relations.
- articulation variant FST: This FST mediates between the phonemic and the phonetic level and incorporates pronunciation variants and inter- and intra-word coarticulation.
- model FST: This minor FST mediates between the phonetic and the acoustic model level. (Here one can state for example that plosives are a composed of a pre-plosive pause model and a plosion model.)

If this "morpho-to-model" FST would be composed with the lattice generated by the phonoacoustic decoder[2], the deep part of the resulting FST would be just the correct lattice for the grammar to parse on, having morphophonemic symbols at each transition. In practice, we use the FST in a different operation mode: We align dynamically the morphophonemic units expected by the language model with the phonoacoustic units delivered by the decoder. This mode of operation relaxes the need to know all phonoacoustic units before being able to start the parse. Like this, the parser and the acoustic decoder can work independently from each other and new information from either side can trigger more parse actions.

Of course, the phonoacoustic decoder should preferably adhere to the restrictions the linguistic part expects. This is just another use of the FSTs. The part of the linguistic knowledge represented by the FST can also be easily used as restriction for the acoustic decoder. The higher levels modeled as Definite-Clause Grammars are not suitable for that but a simple morpheme loop—expressible as FST—can be used instead. So for the experiments described here, such a morpheme-loop-FST was composed with the same morpho-to-model FST mentioned above, the deep symbols were stripped away, giving an FSA modeling a morpheme loop with all pronunciation variants and expressed with HMM-models as symbols. This FSA was minimized and then used as recognition network of the phonoacoustic decoder.

[2]A lattice is formally nothing but a weighted FSA. Simply duplicating each symbol to a symbol pair transforms this FSA to an FST which can be composed to another FST by the FST composition operation

3.2. Experiments

Test and training material were taken from the Swiss-German Polyphone database ([10]). This corpus, which is currently still under construction, contains (finally) speech data from over 5000 speakers recorded over digital phone lines with read utterances from newspapers and spontaneous speech. The tests presented were performed on a selection of 242 utterances of 2 to 32 morphemes (average 9.2), containing cardinal and ordinal numbers, number groups etc. since this is the sub-language the grammar and lexicon of ARCOS are prepared for, representing a very small application domain. However, the phone models were trained on a random selection of the Swiss-German Polyphone so they can be regarded as domain-independent. The alignment phase of the training was itself done using a simple pronunciation variation rule set, thus allowing the recognizer to align with any variation of the standard transcription available.

In order to test the performance of the articulation FST, recognition was done once with the full FST, compiled from two-level rules of some well-known pronunciation phenomena in German, and once with a simple FST, considering only geminate reduction and deletion of the glottal closure, and otherwise copying the phonemes to the corresponding phones.

Used FST	Full	Simple
Morpheme Recognition Rate	85.1%	60.7%
Morpheme Recognition Accuracy	77.1%	53.6%
Utterance Recognition Rate	42.1%	36%

Table 1: Recognition comparison between full and simple articulation FST

Of course, utterance recognition depended heavily on the utterance length:

Used FST	Full	Simple
1-8 morph./utt.	56.9%	43.1%
9-16 morph./utt.	27.9%	14.4%
17-24 morph./utt.	50%	50%
25-32 morph./utt.	0%	0%

Table 2: Whole Utterance Recognition Rate for different utterance lengths

4. LEARNING ARTICULATION VARIATION FSTS FROM EXAMPLES

Articulation variation rules are productive in the sense that to any given phonemic transcription they allow many phonetic realizations. Of course there exists a trade-off in having productive rules. On one hand many realizations are included in the model, but on the other hand, recognition

networks get bigger and confusion between similar utterances can increase if their modeled realizations come too close.

One way to restrict those rules is to weight them according to their frequency in the training set. (See [11]). In the case of FSTs every transition gets a "rule application probability" which can be estimated by the frequency of that transition relative to the times passed at the start state (or "left node").

But it is also important to start with a reasonable set of rules that correspond to the real phenomena encountered in speech. Classical linguistics offers knowledge about the most common phenomena, but there is no systematic collection of rare phenomena, even less in knowledge about probabilities, available. Learning rules automatically (or half automatically) from examples would be here of great advantage.

4.1. The problem of special rules

Manually written FSTs can be kept quite small, since they model general rules known from linguistics. However, trained FSTs tend to become very large, since of all possible contexts to a new phenomenon, the vast majority will never appear in the learning sequence. The rule resulting from contexts that do appear is therefore much too special. The lack of examples leads to a *blowing-up* of the automaton, since many different cases have to be distinguished, which would fall together, had the rule been known to its full generality. Using more and more training data introduces many more new rules rather then completing the set of contexts of each rule.

In the following we will focus on this blowing-up effect problem. The basic approach is to combine an existing small set of hand-made and therefore fairly general rules with a larger amount of automatically learned but therefore incomplete rules. The idea is that the rules of the hand-made set are correct and 'explain' most of the learning sequence and that only the smaller number of remaining isolated examples have to be extracted and used by an expert to find new rules, or to be automatically added to the hand-made rules in an appropriate way.

4.2. Experiments

In this section we describe some experiments we conducted to test how such an "alignment" between a preliminary FST and a reliable two-level sequence can be done and at the same time a new FST can automatically be generated. We do not address the question of how to collect reliable training material for learning new rules.

In the experiments, one particular hand-made automaton, A_0, played the role of the 'real' source. (For that purpose, an automaton was chosen that had served as pronunciation model in previous recognition experiments, but any similar automaton could have done as well.) The learning sequences were generated by a random walk through that automaton. Therefore, the learning sequences did not contain any information other than the information of that automaton, in contrast to a real-life sequence that would have contained other constrains as well!

Of this automaton, two other automata, A_1 and A_2, were derived be simply taking out one of the transitions in one case and all of the transitions consuming a particular symbol in the other. These two automata played the role of the preliminary hand-made automaton. (Hence the transitions taken out from the 'real' automaton played the not yet known phenomena to be learned.) The aim was to find an automaton that was able to explain all old and new phenomena, restrict the new phenomena to the contexts they were found in yet still be of moderate size.

4.2.1. FST-Alignment with Dynamic Programming

The idea here is that the automaton we aim at is a minimal enlargement of the available hand-made automaton. To find such an enlargement, it is tried to fit as much as possible of the sequence to paths within the handmade automaton and allow as little as possible deviations from it. This is done in three steps:

1. First, one generic state (meaning "any not yet known state") is added to the hand-made automaton and a DP finds the best alignment of the sequence with any path of the enlarged automaton. Moves within the old automaton are free, but transitions to and from the additional state are at some cost.
2. Once an optimal alignment path is found, all the extra transitions are added to the original FST at the places they occurred, thus put into the appropriate context, but a new state is generated for every occurrence of the generic state.
3. The non-deterministic automaton resulting from putting all original transitions and all newly created transitions together is determinized and minimalized. By this process, many of the newly generated states are found to be identical.

In some variations to the experiment, also new transitions, between old states but with new symbols, were allowed at some cost, and the cost for exiting the old FST earlier or reentering it later was reduced to award specialization of the new phenomena to more of the context they were found in.

Table 3 shows some results for automaton A_1. The resulting automaton size is given as No.-of-States/No.-Of-Transitions. The goal-FST, A_0, has the dimensions 42/799.

Sequence length	10^3	10^4	42000
both contexts			
encouraged	71/1290	188/3698	415/8142
left context e.			81/1286
right context e.			74/1734
no context e.			112/2723

Table 3: Alignment of training sequences to A_1 by Dynamic Programming.

Although the DP alignment method results in reasonably sized automata, there seem to be at least one major problem with this approach: Every stay at the generic state

has the same cost, so that the alignment will find the optimal path to be the one with the least stays there. But what we want to minimize is the number of new states, no matter how often the path goes through them. This problem goes back to the DP condition that the optimal sub-paths to and from a given state should be determinable independently from each other.

4.2.2. State Assignment (n-m-Gram)

The basic assumption of this approach is that a sufficiently large context can uniquely determine the state within this context. Assuming that the goal automaton corresponds to a given hand-made automaton enlarged by additional states and transitions, and assuming that the basic assumption holds for that given hand-made automaton [3] the algorithm assigns states between each symbol of the sequence. If the corresponding n-m-context is explainable by the original automaton, the state is chosen accordingly, and otherwise, a new state is generated for each new n-m-context. A main difference to the DP-method is that several occurrences of the same n-m-context are assigned the same state in the first place.

Tables 4 and 5 show some results for different training sequence lengths and $n, m = 6, 3$ and $n, m = 4, 2$ respectively.

Sequence length	10^3	10^4	10^5
A_0	42/799	42/799	42/799
A_1	138/2063	868/13457	6067/99643
A_2	50/1006	68/1427	291/6733

Table 4: Alignment to A_1 and A_2 by State Assignment $(6, 3)$

Sequence length	10^3	10^4	10^5
A_0	42/799	42/799	42/799
A_1	107/766	496/9190	2734/52009
A_2	45/901	47/970	72/1812

Table 5: Alignment to A_1 and A_2 by State Assignment $(4, 2)$

4.2.3. State Merging

The goal of the next experiment is to decrease the number of states of an automaton to any desired number with a minimal increase in generality. The idea is to start with any given automaton that can explain a given sequence and reduce its number of states one by one. In each step, one pair of states is selected and merged into one state, while the transitions are merged such that the new automaton still explains the language. (Each such merging increases the generality of the model a bit.) The pair to be merged next is selected such that the decrease in observation probability is minimized:

Since at any step the (non-deterministic) automaton can explain the sequence, the sequence corresponds to a path of that automaton, and hence statistics can be made from which a conditional transition probability can be estimated for each transition. The observation probability of the whole sequence, given that conditional probability distribution, is taken as the value to be maximized.[4]

A somewhat astonishing result emerged when it was tried to determinize and minimize those decreasing non-deterministic automata:

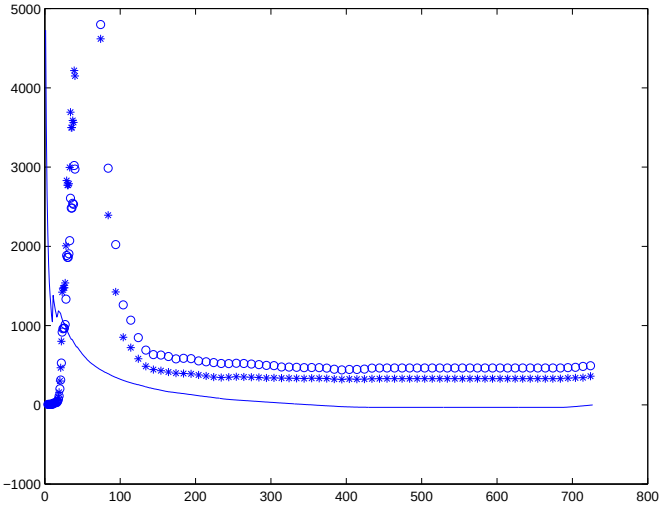

Figure 1: No. of minimal DFA states (circles) and transitions (stars) with decreasing no. of NFA states. (From right to left!) The dotted line shows the negative log of the observation probability of the training sequence.

Obviously at some stage the non-deterministic transducers have deterministic equivalents with a very high number of states. Only further merging of states can bring the DFA dimensions back to reasonable sizes.

5. CONCLUSIONS

It seems worth the effort to cleanly separate different linguistic levels and describe them in separate models. In particular, it is advantageous in our view to regard regular pronunciation variations and inter- and intra-word coarticulation as a relationship between the morphophonemic and the phonetic level.

Finite-State Transducers prove themselves adequate to model simple such regular relations. The FST composition operation allows to merge several models to one for computational efficiency, while the models can be developed and improved separately.

[3] that is, for a context of size n , the state after the $(n-m)$-th symbol is assumed to be uniquely determined, hence the name n-m-gram. For m=0, it is the conventional n-gram assumption.

[4] Actually we want to maximize $p(A|\text{seq})$. But with the daring assumption that $p(A) = p(\text{number-of-states}(A))$ maximizing $p(A|\text{seq})$ is equivalent to maximizing $p(\text{seq}|A)$

Recognition experiments show that a relatively small set of commonly known rules for pronunciation variations and inter- and intra-word coarticulation can significantly improve recognition. If modeled on an intermediate level, they allow the acoustic-phonetic decoding and the morphophonemic lexicon to stay free of those phenomena, such that in an optimal case, only the standard morphophonemic transcription has to be built for each new application domain.

Learning rules from examples — although desirable — poses certain difficulties. In particular, incomplete knowledge of the contexts a phenomenon may appear in leads to complicated rules which demand for large and computationally costly models. Some experiments were presented that hint on how to overcome this problem. Certainly, the proposed alignment of an FST with a trusted two-level sequence can help an expert to find new rules that are relevant to speech recognition.

6. REFERENCES

[1] S. Safra und B. Pfister. ARCOS-G: Ein Experimentalsystem zur Erkennung kontinuierlicher deutscher Sprache. In K. Fellbaum, editor, *Elektronische Sprachsignalverarbeitung*, Studientexte zur Sprachkommunikation, Heft 11, Seiten 174–181, Technische Universität Berlin, Oktober 1994.

[2] K. Koskenniemi. Two-level model for morphological analysis. In *Proc. of the 8th International Joint Conference on Artificial Intelligence*, pages 683–685, 1983.

[3] L. Karttunen, K. Koskenniemi, and R. Kaplan. TWOL: a compiler for two-level phonological rules. In M. Dalrymple, editor, *Tools for Morphological Analysis*. Center for Study of Language and Information, Stanford University, 1987.

[4] G. Ritchie, S. Pulman, A. Black, and G. Russell. A computational framework for lexical description. *Computational Linguistics*, 13(3):290–305, 1987.

[5] Ch. Hoequist and F. Nolan. On an application of phonological knowledge in automatic speech recognition. *Computer, Speech and Language*, 5(5):133–153, 1991.

[6] E. Vidal. Finite-state speech-to-speech translation. In *Proc. ICASSP*, volume 1, pages 111–114. IEEE, 1997.

[7] Ch. Traber. *SVOX: The Implementation of a Text-to-Speech System for German*. Diss. ETH no. 11064. TIK-Schriftenreihe, ETH Zurich, 1995.

[8] L. Karttunen, R. M. Kaplan, and A. Zaenen. Two level morphology with composition. In *Coling-92*, 1992.

[9] K. Huber. Dokumentation: *FSTCompose, ein Programm zur Transformation von Regulären Ausdrücken in FSTs*. TIK, ETH Zürich, Dezember 96.

[10] K. Huber. Swiss German Polyphone: Schlussbericht. TIK, ETH Zürich, April 1998.

[11] G. Lehtinen and S. Safra. Generation and selection of pronunciation variants for a flexible word recognizer. In *this volume*, 1998.