# Re-Engineering Letter-to-Sound Rules

**Martin Jansche**
The Ohio State University
Columbus, OH 43210, U.S.A.
jansche.1@osu.edu

## Abstract

Using finite-state automata for the text analysis component in a text-to-speech system is problematic in several respects: the rewrite rules from which the automata are compiled are difficult to write and maintain, and the resulting automata can become very large and therefore inefficient. Converting the knowledge represented explicitly in rewrite rules into a more efficient format is difficult. We take an indirect route, learning an efficient decision tree representation from data and tapping information contained in existing rewrite rules, which increases performance compared to learning exclusively from a pronunciation lexicon.

## 1 Introduction

Text-to-speech (TTS) systems, like any other piece of sophisticated software, suffer from the shortcomings of the traditional software development process. Highly skilled developers are a costly resource, the complexity and sheer size of the code involved are difficult to manage. A paradigmatic example of this is the letter-to-sound component within the text analysis module of a mature large-scale text-to-speech system. In the system described in (Sproat, 1998) text analysis is performed using finite-state transducers compiled from rewrite rules (Kaplan and Kay, 1994; Mohri and Sproat, 1996) and other high-level descriptions. While the exclusive use of finite-state technology has advantages, it is not without its shortcomings, both technical and stemming from the use of hand-crafted rule sets and how they are represented:

1. Extensive rule sets need to be constructed by human experts, which is labor-intensive and expensive (Sproat et al., 1998).
2. Realistic rule sets are difficult to maintain because of complex interactions between serially composed rules.
3. Although rewrite rules can, in principle, be compiled into a huge monolithic transducer that is then very time-efficient, in practice this is not feasible because of the enormous sizes of the resulting machines (cf. the numbers given in (Mohri and Sproat, 1996) and (Sproat et al., 1998, 74)).
4. For reasons of space efficiency, certain computations are deferred until run-time (Mohri et al., 1996; Mohri et al., 2000), with a significant impact on time efficiency.

While there is a clear need for human expert knowledge (Sproat et al., 1998, 75ff.), those experts should not have to deal with the performance aspects of the knowledge representation. Ideally we would like to use a knowledge representation that is both time and space efficient and can be constructed automatically from individually meaningful features supplied by human experts. For practical reasons we have to be content with methods that address the efficiency issues and can make use of explicitly represented knowledge from legacy systems, so that moving to a new way of building TTS systems does not entail starting over from scratch.

As a case study of how this transition might be achieved we took the letter-to-phoneme rules for French in the TTS system described in (Sproat, 1998) and proceeded to

1. Construct a lexicon using the existing system.
2. Produce an alignment for that lexicon.
3. Convert the aligned lexicon into training instances for an automatically induced classifier.
4. Train and evaluate decision trees.

By running the existing system on a small newspaper corpus (ca. 1M words of newspaper text from *Le Monde*) and eliminating abbreviations we obtained a lexicon of about 18k words. This means that the performance of the automatically trained system built from this lexicon is relative to the existing system.

The key steps, aligning the lexicon and building a training set, are described in detail in Sections 2 and 3 below.

Our choice of decision trees was motivated by their following desirable properties:

1. Space and time efficiency, provided the feature functions can be represented and computed efficiently, which they can be in our case.
2. Generality.
3. Symbolic representation that can easily be inspected and converted.

The first property addresses the efficiency requirements stated above: if every feature function can be computed in time $\mathrm{O}(f)$, where the function $f$ does not involve the height of the decision tree $h$, then the classification function represented by the decision tree can be computed in time $\mathrm{O}(\lambda n.\, h \times f(n)) = \mathrm{O}(f)$ if feature values can be mapped to child nodes in constant time, e. g. through hashing; and similarly for space.

The other properties justify the use of decision trees as a knowledge representation format. In particular, decision trees can be converted into implicational rules that an expert could inspect and can in principle be compiled back into finite-state machines (Sproat and Riley, 1996), although that would re-introduce the original efficiency problems. On the other hand, finite-state transducers have the advantage of being invertible, which can be exploited e. g. for testing hand-crafted rule sets.

We use a standard decision tree learner (Quinlan, 1993), since we believe that it would be premature to investigate the implications of different choices of machine learning algorithms while the fundamental question of what any such algorithm should use as training data is still open. This topic is explored further in Section 5. Related work is discussed in Section 6.

## 2 Aligning the Lexicon

Learning a mapping between sets of strings is difficult unless the task is suitably restricted or additional supervision is provided. Aligning the lexicon allows us to transform the learning task into a classification task to which standard machine learning techniques can be applied.

Given a lexical entry we ideally would want to align each letter with zero or more phonemes in a way that minimizes the descriptions of the function performing the mapping and of the exceptions. Since we do not know how to do this efficiently, we chose to be content with an alignment produced by the first phase of the algorithm described in (Luk and Damper, 1996): we treat the strings to be aligned as bags of symbols, count all possible combinations, and use this to estimate the parameters for a zeroth-order Markov model.

|  |  |  |
|---|---|---|
| (a) | t e x . t e | (b) | t e x t e . . . . . |
|  | t E k s t . |  | . . . . . t E k s t |

Figure 1: Two possible alignments

Figure 1 shows two examples of an alignment, where the dot represents the empty string (for reasons of visual clarity), also referred to as $\varepsilon$. Alignment (b), while not as intuitively plausible as alignment (a), is possible as an extreme case. In general, when counting the combinations of $\ell$ letters with $p$ phonemes, we want to include $p$ empty letters and $\ell$ empty phonemes. For example, given the letters ‘texte’ and corresponding phonemes /tEkst/, we count $C_L(\mathrm{t}, \varepsilon) = 10$, $C_L(\mathrm{t}, \mathrm{t}) = 4$, $C_L(\mathrm{t}, \mathrm{k}) = 2$, etc. By normalizing the counts we arrive at an empirical joint probability distribution $\tilde{P}_L$ for the lexicon.

The existing rewrite rules were another source of information. A rewrite rule is of the form

$$\phi \to \psi \,/\, \lambda \_ \rho$$

where $\phi$ is usually a string of letters and $\psi$ a string of phonemes. The contextual restrictions expressed by $\lambda$ and $\rho$ will be ignored. Typically $\phi$ and $\psi$ are very short, rarely consisting of more than four symbols. We created a second lexicon consisting of around 200 pairs $\langle \phi, \psi \rangle$ mentioned in the rewrite rules, and applied the same procedure as before to obtain counts $C_R$ and from those a joint probability distribution $\tilde{P}_R$.

The two empirical distributions were combined and smoothed by linear interpolation with a uniform distribution $P_U$:

$$P(x, y) = \lambda_1 \tilde{P}_R(x, y) + \lambda_2 \tilde{P}_L(x, y) + \lambda_3 P_U(x, y)$$

where each $\lambda_i \geq 0$ and $\lambda_1 + \lambda_2 + \lambda_3 = 1$. The effects of using different coefficient vectors $\vec{\lambda}$ will be discussed in Section 4.

Since we had available a library for manipulating weighted automata (Mohri et al., 2000), the alignments were computed by using negative log probabilities as weights for a transducer with a single state (hence equivalent to a zeroth-order Markov model), composing on the left with the letter string and on the right with the phoneme string, and finding the best path (Searls and Murphy, 1995; Mohri et al., 2000). This amounts to inserting $\varepsilon$-symbols into both the string of letters and the string of phonemes in a way that minimizes the overall weight of the transduction, i. e. maximizes the probability of the alignment with respect to the model.

## 3 Building Training Instances

Now we bring in additional restrictions that allow us to express the task of finding a function that maps letter sequences to phoneme sequences as the simpler task of inducing a mapping from a single letter to a single phoneme. This is a standard classification task, and once we have a set of feature functions and training instances we can choose from a multitude of learning algorithms and target representations. However, investigating the implications of different choices is not our goal.

The first simplifying assumption is to pretend that translating an entire text amounts to translating each word in isolation (but see the discussion of *liaison* in Section 5 below). Secondly we make use of the fact that the pronunciation of a letter is in most cases fully determined by its local context, much more so in French (Laporte, 1997) than in English.

Each letter is to be mapped to a phoneme, or the empty string $\varepsilon$, in the case of “silent” letters (deletions). An additional mechanism is needed for those cases where a letter corresponds to more than one phoneme (insertions), e. g. the letter ‘x’ corresponding to the phonemes /ks/ in Figure 2a. The problem is the non-uniform appearance of an explicit empty string symbol that allows for insertions. We avoided having to build a separate classifier to predict these insertion points (see (Riley, 1991) in the context of pronunciation modeling) by simply pretending that an explicit empty string is present before each letter and after the last letter. This is illustrated in Figure 2b. Visual inspection of several aligned lexica revealed that at most one empty string symbol is needed between any two letters.

| | | | |
|---|---|---|---|
| (a) | t e x . t e | (b) | . t . e . x . t . e . |
| | t E k s t . | | . t . E . k s t . . . |

Figure 2: Padding aligned strings

From these aligned and padded strings we derived training instances by considering local windows of a fixed size. A context of size one requires a window of size three, which is centered on the letter aligned with the target phoneme. Figure 3 shows the first few training instances derived from the example in Figure 2b above. The beginning and end of the string are marked with a special symbol. Note that the empty string symbol only appears in the center of the window, never in the contextual part, where it would not convey any information.

| | | |
|---|---|---|
| $ . t | $\mapsto$ | . |
| $ t e | $\mapsto$ | t |
| t . e | $\mapsto$ | . |
| t e x | $\mapsto$ | E |
| e . x | $\mapsto$ | . |
| e x t | $\mapsto$ | k |
| x . t | $\mapsto$ | s |
| x t e | $\mapsto$ | t |

Figure 3: A few training instances (context size: 1)

## 4 Evaluation

We delineated a 90%/10% split of the lexicon and performed the alignment using a probability distribution with coefficients $\lambda_1 = 0$, $\lambda_2 = 0.9$, and $\lambda_3 = 0.1$, i. e., no information from the rewrite rules was used and the empirical probabilities derived from the lexicon were smoothed slightly. The value for $\lambda_3$ was determined empirically after several trial runs on a held-out portion. We then generated training instances as described in the previous section, and set aside the 10% we had earmarked earlier for testing purposes. We ran C4.5 on the remaining portion of the data, using the held out 10% for testing. Table 1 summarizes the following aspects of the performance of the induced decision tree classifiers on the test data relative to the size of context used for classification: classification accuracy per symbol; micro-averaged precision (P) and recall (R) per symbol; size of the tree in number of nodes; and size of the saved tree data in kilobytes. All trees were pruned and the subsetting option of C4.5 was used to further reduce the size of the trees.

Further increasing the context size did not result in better performance. We did see a performance in-

| context | acc. | P | R | size of tree | |
|---|---|---|---|---|---|
| letters | % | % | % | nodes | kB |
| 0 | 84.0 | 51.9 | 86.6 | 44 | 7 |
| 1 | 96.6 | 90.0 | 91.3 | 917 | 149 |
| 2 | 98.6 | 97.0 | 97.1 | 2664 | 435 |
| 3 | 98.7 | 97.5 | 97.4 | 3585 | 586 |

Table 1: Performance relative to context size, alignment based on lexicon

crease, however, when we repeated the above procedure with different coefficients $\vec{\lambda}$. This time we set $\lambda_1 = 0.9$, $\lambda_2 = 0.09$, and $\lambda_3 = 0.01$. These particular values were again determined empirically. The important thing to note is that the information from the rewrite rules is now dominant, as compared to before when it was completely absent. The effect this had on performance is summarized in Table 2 for three letters of context. As before, classification accuracy is given on a per-symbol basis; average accuracy per word is around 85%. Notice that the size of the tree decreases as a result of a better alignment.

| alignment | acc. | P | R | size of tree | |
|---|---|---|---|---|---|
| | % | % | % | nodes | kB |
| lexicon | 98.7 | 97.5 | 97.4 | 3585 | 586 |
| lex. + rules | 98.9 | 97.8 | 97.9 | 3394 | 555 |

Table 2: Performance relative to alignment quality (context size: 3)

These figures are all relative to our existing system. What is most important to us are the vast improvements in efficiency: the decision trees take up less than 10% of the space of the original letter-to-phoneme component, which weighs in at 6.7 MB total with composition deferred until runtime, since off-line composition would have resulted in an impractically large machine. The size of the original component could be reduced through the use of compression techniques (Kiraz, 1999), which would lead to an additional run-time overhead.

Classification speed of the decision trees is on the order of several thousand letters per second (depending on platform details), which is many times faster than the existing system. The exact details of a speed comparison depend heavily on platform issues and what one considers to be the average case, but a conservative estimate places the speedup at a factor of 20 or more.

## 5 Directions for Further Research

The tremendous gains in efficiency will enable us to investigate the use of additional processing modules that are not included in the existing system because they would have pushed performance below an acceptable bound. For example no sophisticated part-of-speech (POS) disambiguation is done at the moment, but would be needed to distinguish, e. g., between different pronunciations of French words ending in *-ent*, which could be verbs, nouns, adverbs, etc. The need for POS disambiguation is even clearer for languages with “deep” orthographies, such as English. In conjunction with shallow parsing, POS disambiguation would give us enough information to deal with most cases of *liaison*, an inter-word phenomenon that required special attention in the existing system and that we have so far ignored in the new approach because of the exclusive focus on regularities at the level of isolated words.

We have been using the existing automaton-based system as our baseline, which is unfair because that system makes mistakes which could very well obscure some regularities the inductive approach might otherwise have discovered. Future comparisons should use an independent gold standard, such as a large dictionary, to evaluate and compare both approaches. The advantage of using the existing system instead of a dictionary is that we could generate large amounts of training data from corpora.

But even with plenty of training data available, the paradigms of verbal inflections, for example, are quite extensive in French, inflected verb forms are typically not listed in a dictionary, and we cannot guarantee that sufficiently many forms appear in a corpus to guarantee full coverage. In this case it would make sense to use a hybrid approach that reuses the explicit representations of verbal inflections from the existing system.

More importantly, having more training data available for use with our new approach would only help to a small extent. Though more and/or cleaner data would possibly result in better alignments, we do not expect to find vast improvements unless the restriction imposed by the zeroth-order Markov assumption used for alignment is dropped, which could easily be done. However, it is not clear that using a bigram or trigram model for alignment would optimize the alignment in such a way that the decision tree classifier learned from the aligned data is as small and accurate as possible.

This points to a fundamental shortcoming of the usual two-step procedure, which we followed here: the goodness of an alignment performed in the first step should be determined by the impact it has on producing an optimal classifier that is induced in the second step. However, there is no provision for feedback from the second step to the first step. For this a different setup would be needed that would discover an optimal alignment and classifier at the same time. This, to us, is one of the key research questions yet to be addressed in learning letter-to-sound rules, since the quality of an alignment and hence the training data for a classifier learner is essential for ensuring satisfactory performance of the induced classifier. The question of which classifier (learner) to use is secondary and not necessarily specific to the task of learning letter–sound correspondences.

## 6 Relation to Existing Research

The problem of letter-to-sound conversion is very similar to the problem of modeling pronunciation variation, or phonetic/phonological modeling (Miller, 1998). For pronunciation modeling where alternative pronunciations are generated from known forms one can use standard similarity metrics for strings (Hamming distance, Levenshtein distance, etc.), which are not meaningful for mappings between sequences over dissimilar alphabets, such as letter-to-phoneme mappings.

General techniques for letter-to-phoneme conversion need to go beyond dictionary lookups and should be able to handle all possible written word forms. Since the general problem of learning regular mappings between regular languages is intractable because of the vast hypothesis space, all existing research on automatic methods has imposed restrictions on the class of target functions. In almost all cases, this paper included, one only considers functions that are local in the sense that only a fixed amount of context is relevant for mapping a letter to a phoneme.

One exception to this is (Gildea and Jurafsky, 1995), where the target function space are the subsequential transducers, for which a limit-identification algorithm exists (Oncina et al., 1993). However, without additional guidance, that algorithm cannot be directly applied to the phonetic modeling task due to data sparseness and/or lack of sufficient bias (Gildea and Jurafsky, 1995). We would argue that the lack of locality restrictions is at the root of the convergence problems for that approach.

Our approach effectively restricts the hypothesis space even further to include only the $k$-local (or strictly $k$-testable) sequential transducers, where a classification decision is made deterministically and based on a fixed amount of context. We consider this to be a good target since we would like the letter-to-sound mapping to be a function (every piece of text has exactly one contextually appropriate phonetic realization) and to be deterministically computable without involving any kind of search. Locality gives us enough bias for efficiently learning classifiers with good performance. Since we are dealing with a restricted subclass of finite-state transducers, our approach is, at a theoretical level, fully consistent with the claim in (Sproat, 2000) that letter–phoneme correspondences can be expressed as regular relations. However, it must be stressed that just because something is finite-state does not mean it should be implemented directly as a finite-state automaton.

Other machine learning approaches employ essentially the same locality restrictions. Different learning algorithms can be used, including Artificial neural networks (Sejnowski and Rosenberg, 1987; Miller, 1998), decision tree learners (Black et al., 1998), memory-based learners and hybrid symbolic approaches (Van den Bosch and Daelemans, 1993; Daelemans and van den Bosch, 1997), or Markov models. Out of these the approach in (Black et al., 1998) is most similar to ours, but it presupposes that phoneme strings are never longer than the corresponding letter strings, which is mostly true, but has systematic exceptions, e. g. ‘exact’ in English or French. English has many more exceptions that do not involve the letter ‘x’, such as ‘cubism’ (/kju-bIz@m/ according to `cmudict.0.6`) or ‘mutualsim’.

The problem of finding a good alignment has not received its due attention in the literature. Work on multiple alignments in computational biology cannot be adapted directly because the letter-to-sound mapping is between dissimilar alphabets. The alignment problem in statistical machine translation (Brown et al., 1990) is too general: long-distance displacement of large chunks of material may occur frequently when translating whole sentences, but are unlikely to play any role for the letter-to-sound mapping, though local reorderings do occur (Sproat, 2000). Ad hoc figures of merit for alignments (Daelemans and van den Bosch, 1997) or hand-corrected alignments (Black et al., 1998) might give good results in practice, but do not get us any closer to a principled solution. The present work is another step towards obtaining better alignments by exploiting easily available knowledge in a systematic fashion.

## 7 Conclusion

We presented a method for building efficient letter-to-sound rules from information extractable from, or with the help of, existing hand-crafted rewrite rules. Using decision trees as the new target representation, significant improvements in time and space efficiency could be achieved at the cost of a reduction in accuracy. Our approach relies on finding an alignment between strings of letters and phonemes. We identified a way to improve alignments and argued that finding a good alignment is crucial for success and should receive more attention.

## Acknowledgments

The work reported on here was carried out within Lucent Technologies' Summer Research Program. I would like to thank the people at Bell Labs for their help and support, especially Gerald Penn, who was my mentor for the summer, and Evelyne Tzoukermann. Thanks also to Chris Brew, Gerald Penn, Richard Sproat, and three anonymous reviewers for valuable feedback on this paper. The usual disclaimers apply.

## References

Alan W. Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *Proc. of the 3rd ESCA Workshop on Speech Synthesis*, pages 77–80.

Antal van den Bosch and Walter Daelemans. 1993. Data-oriented methods for grapheme-to-phoneme conversion. In *Proc. of the 6th European Conference of the Association for Computational Linguistics*, pages 45–53.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Rossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Walter M. P. Daelemans and Antal P. J. van den Bosch. 1997. Language-independent data-oriented grapheme-to-phoneme conversion. In Jan P. H. van Santen, Richard W. Sproat, Joseph P. Olive, and Julia Hirschberg, editors, *Progress in Speech Synthesis*, pages 77–89. Springer, New York.

Dan Gildea and Dan Jurafsky. 1995. Automatic induction of finite state transducers for simple phonological rules. In *Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 9–15.

Ronald M. Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378.

George Anton Kiraz. 1999. Compressed storage of sparse finite-state transducers. In *Proc. of the 1999 Workshop on Implementing Automata*, Potsdam, Germany.

Éric Laporte. 1997. Rational transductions for phonetic conversion and phonology. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*, chapter 14, pages 407–430. MIT Press, Cambridge.

Robert Luk and Robert Damper. 1996. Stochastic phonographic transduction for English. *Computer Speech and Language*, 10:133–153.

Corey Andrew Miller. 1998. *Pronunciation Modeling in Speech Synthesis*. Ph.D. thesis, University of Pennsylvania.

Mehryar Mohri and Richard Sproat. 1996. An efficient compiler for weighted rewrite rules. In *Proc. of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 231–238.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 1996. Weighted automata in text and speech processing. In *Extended Finite State Models of Language: Proc. of the ECAI '96 Workshop*, pages 46–50.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32.

José Oncina, Pedro García, and Enrique Vidal. 1993. Learning subsequential transducers for pattern recognition and interpretation tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):448–458.

J. R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Michael D. Riley. 1991. A statistical model for generating pronunciation networks. In *Proc.*

*of the International Conference on Acoustics, Speech, and Signal Processing*, pages 737–740.

David B. Searls and Kevin P. Murphy. 1995. Automata-theoretic models of mutation and alignment. In *International Conference on Intelligent Systems in Molecular Biology*, pages 341–349.

T. J. Sejnowski and C. R. Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168.

Richard Sproat and Michael Riley. 1996. Compilation of weighted finite-state transducers from decision trees. In *Proc. of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 215–222.

Richard Sproat, Bernd Möbius, Kazuaki Maeda, and Evelyne Tzoukermann. 1998. Multilingual text analysis. In (Sproat, 1998), chapter 3, pages 31–87.

Richard Sproat, editor. 1998. *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*. Kluwer, Dordrecht.

Richard Sproat. 2000. *A Computational Theory of Writing Systems*. Cambridge University Press, Cambridge.