Multilingual Data-Driven Pronunciation

R. I. Damper †, Y. Marchand ‡, C. R. Adsett ‡, T. Soonklang †, J.-D. S. Marsters †

† School of Electronics and Computer Science,
University of Southampton, Southampton SO17 1BJ, UK.

‡ Institute for Biodiagnostics (Atlantic),
National Research Council Canada, Neuroimaging Research Laboratory,
1796 Summer Street, Suite 3900, Halifax, Nova Scotia, Canada B3H 3A7.

† rid@ecs.soton.ac.uk, ‡ yannick.marchand@nrc-cnrc.gc.ca

Abstract

Automatic pronunciation of unknown words is a hard problem of great importance in speech technology. The difficulty of the problem appears to vary across languages, according to the 'depth' of orthography, although there are, as yet, little by way of quantitative comparisons. Early solutions were based on manually-written expert rules but these are expensive to derive and maintain, are entirely language-specific and—according to more recent evaluations using large test sets—do not perform very well (at least for English). By contrast, data-driven methods (which infer pronunciations from a set of examples) require little expert knowledge, can be quickly and easily applied to new languages for which an appropriate set of examples is available, and seem to perform far better than rules. In this paper, we compare the success of data-driven automatic pronunciation for four European languages: English, French, Frisian and German. We also attempt to quantify the difficulty of the problem using the entropy of the association between letters and phonemes found after prior alignment of the various dictionaries.

1. Introduction

Text-to-speech (TTS) synthesis has developed enormously over the past decade, and is starting to find many commercial applications. The technology is critically dependent upon the ability to derive an accurate pronunciation for every word in the input. Although a large dictionary, augmented by morphological analysis, can give access to high-quality pronunciations, it is not a complete solution. The set of all possible words is not closed because the infinite generativity of natural language is reflected in neologisms that are outside the set constructed at any particular time. Hence, some secondary or 'back-up' strategy is required to supplement the TTS system dictionary.

By far the most popular way to do this has been for a linguist or phonetician, expert in the particular language of interest, to write manually letter-to-phoneme transcription rules that are supposed to capture the expert's knowledge of the regularities governing the correspondence between writing and sound systems. Representative rule-sets can be found in [1] for English and [2] for English and for French. However, this approach is expensive and time-consuming, the resulting rules are difficult to maintain, and the end result is entirely language-specific. Another serious concern is that recent evaluations using large test sets [3] show typical rules to perform very poorly (at least for English) relative to data-driven techniques [4] which infer pronunciations from a set of examples. By contrast, data-driven methods require little expert knowledge, can be quickly and easily applied to new languages provided an appropriate set of examples is available, and seem to perform far better than rules—although there is some concern that the errors generated are often quite unlike human errors (e.g., vowel-less transcriptions are relatively common in data-driven approaches but rare in rule-derived pronunciations).

In this paper, we compare three data-driven methods for automatic pronunciation on four different European languages: English, French, Frisian and German. The methods studied are pronunciation by analogy (PbA) as described by Marchand and Damper [5], the table look-up method described by Weijters [6], and the table look-up method by Daelemans and van den Bosch [7]. These methods were selected as, first, we believe PbA to be the best currently-available technique for pronunciation of common words [3] and, second, they all belong to the class of 'lazy' learning methods [8, 9] that (unlike Sejnowski and Rosenberg's well-known NETtalk neural network [10], for instance) minimise the requirement for protracted prior training. This means that there is no need to divide the available examples into disjoint training and held-out test datasets. In 'pure' lazy learning (as exemplified by PbA), virtually the entire set of examples in the dictionary of a language is available for testing.

A major concern in this work is how the difficulty of letter-to-phoneme conversion varies across the languages studied. English is especially notorious for the lack of regularity in its spelling-to-sound correspondence. That is, it has a *deep* orthography [11, 12, 13] as opposed to the *shallow* orthography of, for example, Serbo-Croatian [14]. Although the success of automatic letter-to-phoneme conversion is one way that we might quantify the difficulty of the problem and, thereby, the depth of orthography, given the rather loose definition of orthographic depth, the complexity of letter-sound correspondence in many (most?) languages, etc., it is unlikely that any single measure will give an entirely adequate quantification. Hence, we will also use the entropy of the association between letters and phonemes found during prior alignment (see below). This is similar to the approach of van den Bosch et al. [15] who quantified depth of orthography as a two-dimensional measure involving the complexity of letter-phoneme alignment and the complexity of letter-phoneme correspondences.

2. Dictionaries, alignment and entropy

Details of the four dictionaries used in this work as the basis of data-driven inference of pronunciations are given in Table 1. In

Language	Dictionary	Number of ...		
		Letters	Phonemes	Words
English	Webster	26	51	19596
French	Brulex	40	39	25810
Frisian	FHWLEX	39	86	59251
German	Celex	31	59	48486

Table 1: Languages studied and details of the dictionaries used for each one.

each case, the number of letters (i.e., including accented symbols) used to specify word spellings, the number of phonemes used to specify word pronunciations, and the total number of words are specified. Needless to say, the total number of words varies between the languages/dictionaries. This is likely to have an unpredictable effect on our results, since we expect pronunciation performance to vary with the size of dictionary but, at this stage, we know very little about this variation. Further, as each dictionary was compiled independently, it is unknown how representative each one is of, for example, rare words in that language and/or words with relatively unusual pronunciations. Finally, individual decisions will have been made by the dictionary compilers concerning the detail of transcription on the narrow/broad continuum.

Data-driven approaches to letter-to-phoneme conversion generally require the letters of each word in the dictionary to be aligned with the corresponding phonemes in one-to-one fashion (strictly, a bijection), so converting the problem of transduction into one of classification. For the three techniques compared here, the algorithm described in [16] was used for alignment except for the English (Webster) dictionary, which had been manually aligned by Sejnowski and Rosenberg [10].

The algorithm is iterative, and works by successively improving estimates of the probability $P(l, p)$ of a letter l associating with a phoneme p when all words in the dictionary are correctly aligned. An estimate $P_k(l, p)$ of the association probability at the kth iteration is found by frequency counts among the (imperfectly) aligned letters and phonemes. This imperfect alignment is improved at the $(k+1)$th iteration and improved estimates made of the association probabilities until convergence at iteration K (typically 5 or 6).

At any iteration, the (negative) entropy of the association matrix can be calculated as:

$$E_k = -\sum_{l=1}^{N_L} \sum_{p=1}^{N_P} P_k(l, p) \log_2 P_k(l, p) \text{ bits} \qquad (1)$$

where N_L and N_P are the numbers of letter and phoneme symbols (types) in the dictionary, respectively. This is a measure of the 'disorder' of the association matrix and, hence, of the complexity of letter-phoneme correspondence. We take E_K (i.e., at convergence) to be an indicator of orthographic depth.

One complication in determining the entropy is that null symbols are introduced in order for the alignment to be a bijection, and it is uncertain if these should be included in N_L and N_P in eqn. (1). In this work, they have been included.

Table 2 shows the entropy of association for the four dictionaries. Values are shown at initialisation and at convergence of the alignment algorithm, except in the case of the English dictionary that was manually aligned. (See [16] for details of initialisation; this was the so-called naïve initialisation that effectively counts letters and phonemes appearing in the same word as associations.) English has (marginally) the highest entropy and German has the lowest; on this basis, we would predict the best pronunciation accuracy for German and the lowest for English.

Language	Dictionary	Entropy (bits)	
		Initial	Converg.
English	Webster	5.696*	5.696*
French	Brulex	8.760	5.606
Frisian	FHWLEX	9.126	5.659
German	Celex	9.012	5.406

Table 2: Entropy of the association matrix at initialisation and at convergence of the alignment algorithm for the languages studied. *As Webster's dictionary was aligned manually, the 'initial' and 'converged' values are shown as identical.

3. Overview of the techniques

The three data-driven methods of automatic pronunciation used in this study will now be briefly described. Note that, for simplicity, we have ignored stress assignment in this evaluation. We also ignore words with null letters when assessing pronunciation performance, since these would obviously never be present in the input to a real TTS system.

3.1. Pronunciation by analogy

The assumption underlying PbA is that the dictionary contains implicit phonological knowledge which can be exploited to generate a pronunciation for an unknown word. An early and influential PbA system was PRONOUNCE [17], and many variants have since been based on it (e.g., [18, 19, 20]). The particular variant used here is that of Marchand and Damper [5].

When an unknown word is presented as an input to the system, so-called full pattern matching between the input letter string and dictionary entries is performed, starting with the initial letter of the input string aligned with the end letter of the dictionary entry. If common letters are found in matching positions in the two strings, their corresponding phonemes (according to the prior alignment) and information about their positions in the input string are used to build a pronunciation lattice, as detailed next. One of the two strings is then shifted relative to the other by one letter and the matching process continued, until the end letter of the input string aligns with the initial letter of the dictionary entry.

The pronunciation lattice is a directed graph that defines possible pronunciations for the input string, built from the matching substring information. A lattice node represents a matched letter, L_i, at some position, i, in the input. The node is labelled with its position i and the phoneme corresponding to L_i in the matched substring, P_{im} say, for the mth matched substring. An arc is labelled with the phonemes intermediate between P_{im} and P_{jm} $(j > i)$ in the phoneme part of the matched substring and the frequency count, increasing by one each time the substring with these phonemes is matched during the search through the lexicon. Arcs are directed from i to j. If the arcs correspond to bigrams, the arcs are labelled only with the frequency. (The string of phonemes intermediate between P_{im} and P_{jm} is empty.) Phonemes P_{im} and P_{jm} label the nodes at each end of the arc, i.e., i and j respectively. Additionally, there is a *Start* node at position 0 and an *End* node at position equal to the length of the input string plus one.

Finally, the decision function identifies the 'best' candidate pronunciation of the input according to some criterion. Possible pronunciations correspond to the string assembled by concatenating the phoneme labels on the nodes or arcs in the order that they are traversed in moving through the lattice from *Start* to *End*. If there is just one candidate corresponding to a unique shortest path, this is selected as the output. If there are tied shortest paths, five different scoring strategies are applied and the winning candidate selected on the basis of their rank [21].

3.2. Table look-up I (TLU I)

This method was proposed by Weijters [6] who argued that his simple look-up procedure is superior to NETtalk [10]. The first step is to create from a training set a table containing n-grams (strings of n letters, n odd), the corresponding phoneme(s) for the middle letter of each n-gram, and their frequencies in the training data. One n-gram is produced for each letter of the input, with each letter serving in turn as the centre of the n-gram. To obtain the pronunciation for an input string, we search for the *closest-fit* n-grams, i.e., those with the highest matched value between the n-gram of the input string and those of the pre-compiled look-up table. If different n-grams corresponding to different phonemes have the same matched value, the phoneme that is maximally represented among the tied n-grams is selected. Weijters used a range of different values of n and weight sets; results presented later will be for the choices that gave the best results.

After matching, the phonemes of the closest-fit n-grams are concatenated to form the pronunciation. If closest-fit n-grams are tied and correspond to different phonemes, the phoneme that occurs most frequently is chosen. If the frequencies are equal, the first one of the tied phonemes is chosen arbitrarily.

3.3. Table look-up II (TLU II)

Daelemans and van den Bosch [7] describe a method similar to Weijters', but having defaults that are invoked in the case of matching failure. It was hoped that this would improve generalisation ability relative to TLU I. During table construction, all unambiguous one-to-one letter-to-phoneme mappings are found and stored in the 0-1-0 subtable. Then, the width of the letter window is expanded on the right by one character, and all unambiguous 0-1-1 patterns found and stored in the 0-1-1 subtable, excluding those patterns already in the 0-1-0 subtable. Then, the window width is expanded on the left by one character and the procedure repeated. The process of expanding the window on right or left and storing all the patterns that have not been stored in the earlier table continues until all patterns in the training set are compressed in the look-up table. Additionally, two default tables are assembled to provide generalisation ability. The first default table, referred as a *best-guess* table, contains all occurring 1-1-1 patterns and their most frequently occurring phonemic mapping. The second table, referred as a *final-guess* table, contains all letters and their most frequently occurring phonemic mappings.

The conversion algorithm starts by searching for a matching letter pattern for each letter of an input word in the 0-1-0 subtable. If found, this is guaranteed to be unambiguous. If no pattern is matched, each letter is extended to a 0-1-1 pattern and the 0-1-1 subtable is then searched. This is repeated until a matching pattern with a minimal extension is found and the corresponding letter-phoneme mapping is returned. If no matching can be found, the best-guess table is scanned to return the 'best' mapping. If look-up table retrieval fails again, the default phoneme of that letter from the final-guess table is returned. Finally, all phonemes are concatenated to create the pronunciation.

Language	PbA % Words	TLU I % Words	TLU II % Words
English	65.46	55.93	58.10
French	92.46	89.93	90.39
Frisian	85.83	81.43	81.52
German	93.05	88.47	88.12

Table 3: Results of automatic pronunciation for the three data-driven methods across the four languages.

4. Results

The performances of the three methods across the four languages were evaluated using a leave-one-out strategy, also known as k-fold cross validation. That is, in the case of PbA, each word was removed in turn from the dictionary and a pronunciation derived by analogy with the remaining words. In the case of table look-up, a note was made during table compilation of those n-grams which were unique to a particular word and that word was stored with the n-gram. Then, when finding a pronunciation for a particular word, n-grams unique to that word were removed from the look-up table. For TLU I, frequencies were recomputed to account for the removal of the test word.

Results were obtained by scoring the automatically-derived pronunciations against the dictionary pronunciations, in terms of words correct. We have previously argued that words correct is a more stringent measure of pronunciation accuracy and should be used in strong preference to phonemes correct [3] so only these figures are given here. Table 3 shows the results obtained. The best TLU I results were obtained for the following weight sets:

English: 9-gram, 16-64-256-1024-256-64-16-4-1

French: 7-gram, 1-4-16-64-16-5-1

Frisian: 9-gram, 16-64-256-1024-256-65-16-4-1

German: as for English

As can be seen in the table, PbA achieved easily the highest percentage of words correct across all four languages. (The reader should note that the fact of testing with tens of thousands of words means that differences of just one or two percentage points are highly statistically significant.) With the exception of English where TLU II is superior, there is little to choose between the two table look-up methods.

Figure 1 shows a scatter diagram of word accuracy versus entropy of the association matrix for the three automatic pronunciation methods and four languages. We expected to see a negative correlation between pronunciation accuracy and entropy but it is less obvious than we would have thought (Pearson correlation coefficient $r = -0.6253$ for the pooled data). It seems that, at least as determined in this study, one or both of entropy and automatic pronunciation performance is not a very good measure of orthographic depth.

5. Conclusions

We have compared three data-driven methods for automatic pronunciation for four European languages, namely: pronunciation by analogy (PbA), the table look-up method by Weijters (TLU I), and the table look-up method by Daelemans and

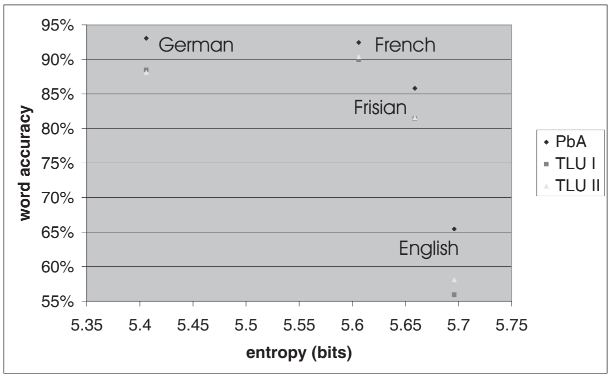

Figure 1: Scatter diagram of word accuracy versus entropy of association matrix for the three automatic pronunciation methods and four languages.

van den Bosch (TLU II). Results confirm a correlation between entropy of letter-phoneme association and difficulty of automatic pronunciation across the deep/shallow orthography continuum but this is weaker than we had expected. Nonetheless, some languages are more difficult to align and transcribe than others. In our work, German is the easiest language in this respect and English is the hardest.

We intend in the future to include stress and syllabification in the evaluation, to study more (and more diverse) languages and, where possible, to use multiple dictionaries for each one. This should give some robustness against the idiosyncrasies of any one particular dictionary. We also wish to correlate the performance of expert rules with the entropy and the performance of the data-driven approaches, as a direct test of the often-seen assertion that rules are adequate for languages with shallow orthography.

6. Acknowledgements

We are grateful to Jelske Dijkstra and the Fryske Akademy for providing the Frisian FHWLEX dictionary for this study.

7. References

[1] H. S. Elovitz, R. Johnson, A. McHugh, and J. E. Shore, “Letter-to-sound rules for automatic translation of English text to phonetics,” *IEEE Transactions on Speech and Audio Processing*, vol. ASSP-24, no. 6, pp. 446–459, 1976.

[2] M. Divay and A. J. Vitale, “Algorithms for grapheme-phoneme translation for English and French: Applications for database searches and speech synthesis,” *Computational Linguistics*, vol. 23, no. 4, pp. 495–523, 1997.

[3] R. I. Damper, Y. Marchand, M. J. Adamson, and K. Gustafson, “Evaluating the pronunciation component of text-to-speech systems for English: A performance comparison of different approaches,” *Computer Speech and Language*, vol. 13, no. 2, pp. 155–176, 1999.

[4] R. I. Damper, Ed., *Data-Driven Methods in Speech Synthesis*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2001.

[5] Y. Marchand and R. I. Damper, “A multistrategy approach to improving pronunciation by analogy,” *Computational Linguistics*, vol. 26, no. 2, pp. 195–219, 2000.

[6] A. Weijters, “A simple look-up procedure superior to NETtalk?” in *Proceedings of International Conference on Artificial Neural Networks (ICANN-91)*, vol. 2, Espoo, Finland, 1991, pp. 1645–1648.

[7] W. Daelemans and A. van den Bosch, “Tabtalk: Reusability in data-oriented grapheme-to-phoneme conversion,” in *Proceedings of 3rd European Conference on Speech Communication and Technology, Eurospeech’93*, vol. 2, Berlin, Germany, 1993, pp. 1459–1466.

[8] D. W. Aha, “Lazy learning,” *Artificial Intelligence Review*, vol. 11, no. 1–5, pp. 7–10, 1997.

[9] A. van den Bosch, A. Weijters, H. J. van den Herik, and W. Daelemans, “When small disjuncts abound, try lazy learning,” in *Proceedings of the 7th Belgian-Dutch Conference on Machine Learning, BENELEARN-97*, Tilburg, The Netherlands, 1997, pp. 109–118.

[10] T. J. Sejnowski and C. R. Rosenberg, “Parallel networks that learn to pronounce English text,” *Complex Systems*, vol. 1, no. 1, pp. 145–168, 1987.

[11] M. Coltheart, “Lexical access in simple reading tasks,” in *Strategies of Information Processing*, G. Underwood, Ed. New York: Academic Press, 1978, pp. 151–216.

[12] I. Liberman, A. Liberman, I. Mattingly, and D. Shankweiler, “Orthography and the beginning reader,” in *Orthography, Reading and Dyslexia*, J. Kavanagh and R. Venezky, Eds. Baltimore, OH: University Park Press, 1980, pp. 137–153.

[13] G. Sampson, *Writing Systems*. London, UK: Hutchinson, 1985.

[14] M. T. Turvey, L. B. Feldman, and G. Lukatela, “The Serbo-Croatian orthography constrains the reader to a phonologically analytic strategy,” in *Orthographies and Reading, Perspectives from Cognitive Psychology, Neuropsychology and Linguistics*, L. Henderson, Ed. London, UK: Lawrence Erlbaum Associates, 1984, pp. 81–89.

[15] A. van den Bosch, A. Content, W. Daelemans, and B. De Gelder, “Measuring the complexity of writing systems,” *Journal of Quantitative Linguistics*, vol. 1, no. 3, pp. 178–188, 1994.

[16] R. I. Damper, Y. Marchand, J.-D. S. Marsters, and A. I. Bazin, “Aligning text and phonemes for speech technology applications using an EM-like algorithm,” *International Journal of Speech Technology*, vol. 8, no. 2, pp. 149–162, 2005.

[17] M. J. Dedina and H. C. Nusbaum, “PRONOUNCE: A program for pronunciation by analogy,” *Computer Speech and Language*, vol. 5, no. 1, pp. 55–64, 1991.

[18] K. P. H. Sullivan and R. I. Damper, “Novel-word pronunciation: A cross-language study,” *Speech Communication*, vol. 13, no. 3–4, pp. 441–452, 1993.

[19] F. Yvon, “Grapheme-to-phoneme conversion using multiple unbounded overlapping chunks,” in *Proceedings of Conference on New Methods in Natural Language Processing (NeMLaP-2’96)*, Ankara, Turkey, 1996, pp. 218–228.

[20] R. I. Damper and J. F. G. Eastmond, “Pronunciation by analogy: Impact of implementational choices on performance,” *Language and Speech*, vol. 40, no. 1, pp. 1–23, 1997.

[21] R. I. Damper and Y. Marchand, “Information fusion approaches to the automatic pronunciation of print by analogy,” 2005, *Information Fusion*, in press.