EFFICIENT LEXICAL RETRIEVAL FOR ENGLISH TEXT-TO-SPEECH SYNTHESIS

Daniel Faulkner & Charles Bryant

Aculab PLC, Lakeside, Bramley Road, Mount Farm, Milton Keynes, MK1 1PT, UK

ABSTRACT

We present a first version of a *filter dictionary* for use in a computer-telephony text-to-speech synthesis system. The aim of the filter dictionary was to provide a lexicon that was compact, fast and had broader coverage than the standard dictionary used to create it. Correct phonemic transcriptions *and* lexical stress assignment were both required for a transcription to be deemed accurate. The approach taken here guarantees 100% accurate coverage of the original dictionary, but also gives 93% accurate transcription of the expected coverage of novel words. Lexical stress and the phonemic transcription were retrieved in one pass, resulting in an extremely fast system. We also allowed user-definition to retain accuracy for non-standard transcriptions. This algorithm was developed for British English, but could be applied to other languages.

1. INTRODUCTION

One of the primary issues of text-to-speech synthesis is accurate grapheme-to-phoneme conversion for English. English does not have a very consistent mapping from spelling to phonology, so the task is more challenging than for a language such as modern Greek, which has a largely one-to-one correspondence.

To tackle this problem, we present a dictionary composed of fragments. Some of these were pre-determined by us, using morphological knowledge [6], others were derived automatically from a lexicon of spellings and their corresponding phonemic representations. We have called the resulting lexicon a *filter dictionary*, because it constructs words from the input text out of the largest fragments it has at its disposal. A result of this is that the coverage of the dictionary is not unlimited, but the benefit is that there is a very high accuracy level in transcription and lexical stress assignment.

To ensure accurate lexical stress assignment as well as correct phonemic transcription, English text-to-speech synthesisers often make use of an on-line dictionary for the majority of their grapheme-to-phoneme conversion. These typically contain machine-readable phonemic representations of a given word, including lexical stress notation, but their coverage is obviously limited to only those words that are listed. If a word in the input text is not contained in the lexicon, the system may access an appendix dictionary and/or a set of letter-to-sound rules (*LTS rules).* LTS rules typically work linearly through a word and map a letter at a time onto a phoneme. Because of the density of English spelling conventions however, such rules are usually inaccurate, especially in assigning lexical stress correctly.

Alternatives to the dictionary + LTS approach include data-driven models of grapheme-to-phoneme conversion. These methods usually involve training algorithms on large, transcribed corpora and deriving statistically likely correspondences between grapheme and phoneme sequences. There are numerous statistical techniques, but many are not within the scope of this paper (e.g. neural networks [5], hidden Markov models [7]), as they are by nature prohibitively large for the memory and processing-time restrictions imposed by our computer-telephony application. In addition, data-driven techniques do not lend themselves well to user-definition of phoneme strings, and we consider this to be crucial in coping with non-standard pronunciations. Finally, there is a fundamental weakness found in some implementations of data driven techniques in grapheme-to-phoneme conversion. Despite the fact that they are often trained on sets of data that do not reflect all the possible letter sequences and corresponding phoneme strings in the target language, they are still applied to unlimited input text once the correspondences have been retrieved. This means that inaccuracies or lack of coverage inherent to the source data are built into the system. This reflects poorly on the algorithm's results, even though it might have dealt perfectly with all sequences it had been trained on.

While we did use an algorithm to extract candidate mappings from a source lexicon, we decided from the outset not to apply the fragments generated from our relatively small set of data to the entire language. This would only cloud the genuine results (i.e. we wanted to know how the dictionary dealt with known and novel words it *should* have been able to cope with). We constructed our dictionary in such a way that - at least in development - it would query letter sequences it had not encountered before. Another difference between our approach and many data-driven techniques is that we carried out some manual editing on the results of the extractions, so that our decisions were not dictated by statistical likelihoods that once again might be corrupted by the limitations of the data.

The approach taken in designing this dictionary was also partly motivated by the theory of pronunciation by analogy [3], in that, where appropriate, use can be made of knowledge of the pronunciations of certain words, in order to create a viable pronunciation for a previously unseen word. It should be noted however, that our method is not proposed as a psychological model for reading aloud, nor do we claim that our system currently provides unlimited coverage.

There are parallels between the filter dictionary and morphological approaches as well [1]. However the filter dictionary does not morphologically deconstruct input words. In addition, while we did make use of morphological analysis, we did not stipulate that all fragments in the dictionary *must* be morphemic (we found that many consistent non-morphemic correspondences exist). We also retained some elements of small-unit and single grapheme-to-phoneme conversion where appropriate, for cases where words could not be fully constructed out of larger fragments. For input words made up entirely of letter sequences that occurred in the original dictionary, these were not expected to be accessed often, as larger fragments should prevent the need for one-to-one mappings.

It should be noted that we stipulated that accuracy of transcription and stress assignment was more important than coverage. LTS rules have unlimited coverage, but their accuracy level is not high. The literature shows that systems that completely replace LTS rules suffer similar problems as their predecessors (e.g. context-driven HMM's do not deliver particularly high accuracy [8]). Prioritising compact flexibility above accuracy seems to have an adverse effect on the system's results. The filter dictionary was built from a relatively small source, so we knew from the outset that it would not have unlimited coverage. According to our requirements though, the filter dictionary must construct an accurate phonemic string that includes accurate lexical stress assignment in one pass (as opposed to [2], for example, where lexical stress is assigned separately from the phoneme string). The prioritisation of accuracy over coverage is not the usual approach but, as stated previously, we view ensuring accurate transcription as being more important than flexibility at this stage of development.

On the basis of these points, we set the following performance targets for our dictionary:

i. Reduced lexicon size in memory.
ii. High level of phonemic transcription accuracy.
iii. High level of lexical stress assignment accuracy.
iv. Increased coverage beyond that of our simple lexicon.
v. User-definition allowed.
vi. Low CPU requirements.

2. BUILDING THE DICTIONARY.

We used the Aculab lexicon as the starting point for our filter dictionary. This contains 60,000 words, each of which has an accurate phonemic transcription and is marked for primary lexical stress. The data was compressed, and the total size in memory was 1.47 MB.

Using a left-to-right, longest match first algorithm (e.g. preferably analyse *ch* as a unit rather than as *c* followed by *h* individually), we aligned the graphemes with their phonemic transcriptions. Lexical stress was not included in the alignment process, because the letters that represent vowels are not inherently stressed or unstressed. When this had been completed, we copied the lexical stress marks from the original dictionary into the newly aligned dictionary. The intention was to incorporate lexical stress assignment straight into the fragments for use in the filter dictionary. This was chosen in preference to using a separate stress-prediction algorithm, as such methods do not seem to yield high success rates [2]. This approach may have resulted in a higher number of fragments than if stress had been ignored, but we felt that it was more natural and efficient to retrieve the lexical stress and the phonemic transcription simultaneously. The advantage of this was that contexts that might dictate stress assignment (e.g. the suffix **/ei1 sh @ n/** always takes lexical stress, regardless of the root's stress conditions) or phonemic alternations (e.g. word final **'s'** may be transcribed as **/z/**, **/iz/** or **/s/**, depending on the final segment of the root) were incorporated directly into the fragments. Therefore, no run-time calculation was required.

To determine correspondences, we firstly used a program to match known inflectional and derivational morphemes [6] to candidate phonemic transcriptions from our aligned lexicon (e.g. we found potential transcriptions for *-al* according to its occurrences in the lexicon). The results were checked by hand, and the transcription that we considered to be the most natural was retained. To avoid incorporating effects of the original dictionary's limitations into the set of fragments, these judgements were based on linguistic knowledge and intuitions rather than frequency alone. The morphemes were then stored as independent units in the dictionary, along with their transcriptions.

Secondly we automatically identified other correspondences. This was done by retrieving frequent alignments from the Aculab lexicon. When the fragment was considered to be reliable (i.e. it would not over-generate incorrect pronunciations or incorrect lexical stress assignment), it was included. This proved to be sufficiently effective that all but 11 of the manual rules could be deleted as they were now either rendered redundant by other fragments, or actually captured as fragments by the algorithm.

The final dictionary provided various mappings from w*ord*-to-phoneme, *morpheme*-to-phoneme, *fragment*-to-phoneme, and *single grapheme*-to-phoneme. Redundant fragments (i.e. fragments that were not used to build themselves and at least one other word) were discarded. When this was completed, we removed from the original lexicon all words that could be successfully transcribed and stressed by the fragments. Exceptions to the fragment correspondences were retained as whole words.

The resulting lexicon contained 24,384 fragments, and occupied 470Kb in memory, a reduction of 59%. Had we not decided that accurate lexical stress assignment should be an intrinsic part of the dictionary, and aimed for accurate phonemic transcription alone, then the size of the dictionary and the number of fragments would be expected to be considerably smaller.

It should be noted that because the fragments had been built from only 60,000 words (and assuming that these did not represent all possible spelling sequences in English), novel letter strings would cause the dictionary to fail. Only word-final or word initial letters received one-to-one correspondences in the fragment generation, so unlimited coverage was not expected. It would be a trivial, but self-defeating task to incorporate LTS rules into the dictionary. Unlimited coverage would be achieved, but this would defeat the object of insisting on a high degree of accuracy, and mask the actual results of the algorithm. This apparent weakness in the dictionary's coverage could be easily - and better - overcome by exposing the algorithm to a larger proportion of the language, because new fragments would be built automatically to cope with novel letter strings. With exposure to a large enough list of aligned spellings and transcriptions, the filter dictionary should be able to build suitable fragments to account for all spellings while maintaining a low memory overhead. In effect, LTS rules are the conceptual opposite of our approach, because they can transcribe any word, but do not have a high accuracy rate.

The dictionary is also user-definable because a word with a specific, non-standard pronunciation could be entered as a fragment, and then derivations could be realistically built from the non-standard pronunciation. An example of this is the pronunciation of the name 'Cholmondley'. If this was entered as a fragment and transcribed as **/ch uh1 m l ii/**, the filter dictionary would be able to generate accurate phonemic transcriptions and lexical stress assignments for potential derivations such as 'Cholmondleyism', 'Cholmondleyite' etc.

3. TESTING THE DICTIONARY

Having established that the filter dictionary could accurately transcribe 100% of the original lexicon while using less space, we wanted to test how it dealt with new words.

We took 100 existing English words from *Chambers English Dictionary* that did not appear in the original Aculab lexicon, and tested them against the filter dictionary. We also devised 100 phonotactically valid pseudo-words (e.g. *sclave*), and surveyed 50 peoples' intuitive pronunciations of them. We made a list of the most frequently proposed pronunciations and tested it against the filter dictionary's output. Finally, we tested the filter dictionary against 100 possible derivations of words that actually did appear in the original lexicon (some of these derivations did not form real words, but the dictionary should be able to transcribe such strings realistically if it has suitable fragments at its disposal).

We expected the filter dictionary to struggle most in transcribing the pseudo-words, because they required it to find correspondences that it had never encountered. As the fragments in the filter dictionary were devised from only the strings found in the Aculab lexicon, many of the pseudo-word letter sequences did not yet exist as fragments, so the dictionary was expected to fail. The genuine word list tended to consist of more common orthographic sequences, so it was expected that the dictionary would more effectively deal with standard words whose letter sequences *had* been incorporated as fragments. We expected the dictionary to cope best with the derivations of words that were contained in the original dictionary, because we knew that most of the fragments would have been encountered before, even though the whole words had never been included.

We measured the CPU requirements of the filter dictionary by timing how long was required to look up and check (against the original lexicon) every word in the filter dictionary (*T*), the time taken to read in all the fragments (*T1*), and how long was required to read in the original dictionary (*T2*). As reading in the fragments would not occur at run-time in the synthesiser, and the original lexicon would not be required at all, processing time was considered to be the T minus T1 minus T2 divided by the whole number of words in the dictionary (N). This would yield the average time to look up a word, build its transcription and check it against the original dictionary. As we did not subtract the time required to check the accuracy of a transcription as well, the CPU time estimate is very conservative; the transcription would not be checked against a control lexicon at run-time:

$$\text{Avg. CPU time} = \frac{\text{T - T1 - T2}}{\text{N}}$$

4. RESULTS

The dictionary transcribed 18% of the pseudo-words, 29% of the unknown *Chambers Dictionary* words, and 64% of the derivations of known words. The dictionary failed when it had to transcribe a letter sequence for which it had no stored fragment. Very little difference would have been made to these results if only accurate phonemic transcription had been accepted. Firstly, lexical stress was an intrinsic part of the fragments, and secondly, it was unknown phoneme sequences that caused the failure rather than incorrect stress.

The transcriptions were accurate for 93% of the words that were successfully constructed from the fragments. A transcription was only deemed to be accurate if its phonemic transcription and its lexical stress assignment were correct.

The average CPU time required to build and check a word's transcription on a 200MHz-Pentium processor was calculated.

$$\frac{4.65\text{s} - 1.55\text{s} - 0.51\text{s}}{60343} = 42.9 \times 10^{-6}\,\text{s}$$

The average time is approximately 43 μs. This is obviously quick enough to maintain real-time processing in a speech synthesis system.

5. DISCUSSION

As expected, the dictionary failed to produce a transcription very frequently for unknown words. It failed if it was presented with a new letter sequence. In terms of coverage, these are not good results, but they are not worrying either, because they were anticipated. As discussed previously, by preventing the dictionary from building transcriptions for fragments it had not previously encountered, we were aware that the coverage would suffer. If the algorithm were exposed to a greater proportion of the language, the resulting dictionary would have a very wide coverage and a very favourable level of accuracy.

The fact that the output was accurate for phonemic transcription and lexical stress for 93% of the words that *were* constructed is highly encouraging when compared with other approaches reported in the literature [2,8].

One important implication of the accuracy of the filter dictionary is that it is possible to obtain both the phonemic string and lexical stress correctly in one pass, as opposed to generating a phoneme string in one pass, and employing a stress-assignment algorithm in a second pass.

Phonemic context was also successfully incorporated directly into the fragments, negating the need for left and right context matching at run-time.

If suitably large lexicons of accurate transcriptions were not available, it would be trivial to incorporate sets of LTS rules into the filter dictionary instead. This would deliver 100% coverage (so that the dictionary could be used in a real system), but the accuracy level would inevitably drop. Likewise, we could have manually included smaller fragments in the filter dictionary. However, by prioritising accuracy above compression, the algorithm exposed the inconsistencies in small unit correspondences, so a similar problem would have arisen as if we had used LTS rules.

The CPU time is very fast and, as the process is still a simple look-up, the benefit to efficiency of excluding run-time calculation (such as context-matching) is clear.

6. CONCLUSION

We aimed to build a dictionary that conformed to the six targets specified in the introduction. All of those targets have been met.

i. We have succeeded in reducing the size of the original lexicon - the filter dictionary is smaller than our previous standard lexicon by 59%.
ii. We have maintained the phonemic accuracy for 100% of the words that occurred in the original dictionary.
iii. We have successfully maintained accurate lexical stress assignment for 100% of the original dictionary.
iv. For unknown words that the dictionary can currently generate, there is an accuracy of 93% for phonemic transcription and lexical stress assignment, so increased coverage has been achieved also.
v. User-definition has been maintained, so any word whose transcription is user-defined can still be employed by the filter dictionary to create accurate derivations.
vi. CPU times are very fast - 1000 times faster than the processing times reported in [2]. Although the processors used by the two systems are different, it is unlikely that a difference of this order is down to the processor alone.

Useful further work would include expanding the coverage of the filter dictionary by training it on a larger proportion of English spellings and transcriptions.

It would also be useful to apply an expanded version of the dictionary to a standard corpus of English words so that it could be compared to other methods (e.g. the ONOMASTICA corpus [8]), and this is something we intend to do shortly.

Although this algorithm was designed to work on English, there is nothing inherent in the system that would prevent it from being used on any other language, as it should be capable of automatically generating suitable fragments for the accurate transcription of most alphabetic writing systems, provided it is exposed to a suitably aligned source of transcribed words.

7. REFERENCES

1. Allen, J. "Synthesis of speech from unrestricted text." *Proceedings of the IEEE, 64.* pp.433-442, 1976.
2. Bagshaw, P. "Phonemic transcription by analogy in text-to-speech synthesis: Novel word pronunciation and lexical compression." *Computer Speech and Language, 12.* pp.119-142, 1998.
3. Glushko, R.J. "Principles for pronouncing print: the psychology of phonology." *Interactive processes in Reading.* pp.61-84, 1981.
4. Landau, S.I and Ramson, W.S. (Eds.) "Chambers English Dictionary, 7th Edition" *Chambers Cambridge,* 1989.
5. Lucas, S.M. and Damper, R.I. "Syntactic neural networks for text-to-phonetics translation." *Proceedings of the International Conference on Acoustics, Speech and Signal Processing, Vol.1.* pp.509-512
6. Marchand, H. "The categories and types of present day English word formation." *Beck, Munchen.* 1969.
7. Parfitt, S.H. and Sharman, R.A. "A bi-directional model of English pronunciation." *Proceedings of the European Conference on Speech Communication Technology (Eurospeech), Vol.2.* pp.801-804, 1991.
8. Yvon, F. "Self-learning techniques for grapheme-to-phoneme conversion." *Onomastica Research Colloquium.* London, 1994.