Comparative Objective and Subjective Evaluation of Three Data-Driven Techniques for Proper Name Pronunciation

Tasanawan Soonklang †, Robert I. Damper †, Yannick Marchand ‡

† 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.

†{ts03r|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. Proper names constitute an especially difficult class of words to pronounce because of their low frequency of occurrence and variable origin. In this paper, we compare three different data-driven approaches which use a dictionary of (known) proper names to infer pronunciations for unknown names, namely: pronunciation by analogy (PbA), the table look-up method described by Weijters, and the ‘improved’ table look-up method by Daelemans and van den Bosch. Evaluation is both objective, in which inferred pronunciations are compared with ‘gold standard’ dictionary pronunciations, and subjective, in which listeners rated synthesised pronunciations using a 5-point scale. Objective evaluation used a leave-one-out technique on 52,911 names in the CMUDICT dictionary. Results show that PbA achieves the best performance at 63.93% names correct. In the subjective evaluation of 529 different pronunciations of 200 names, 12 listeners rated the pronunciations. Non-parametric tests of significance show that the dictionary pronunciations are rated superior to the automatically-inferred pronunciations; PbA is superior to both table look-up methods, and the ‘improved’ table look-up is superior to Weijters’ original method (Walsh test, p < 0.005 in all cases.)

1. Introduction

Text-to-speech (TTS) synthesis technology, which has developed enormously over the past decade, enables a voice interface for many commercial applications, especially in telecommunications. These applications often require the pronunciation of proper names, i.e., the names of people, streets, cities, places, companies, etc. The automatic pronunciation of proper names remains a real challenge for text-to-phoneme conversion, a component part of the pronunciation module in TTS synthesis, because the geographical and language origin of the names can be varied, and the number of distinct names is very large [1, 2].

Numerous studies have proposed various approaches to the problem of proper name pronunciation and have reported the results in different ways. Most studies evaluate the performance objectively in terms of word accuracy; by comparing the pronunciations generated by their models with those in some data set taken as a ‘gold standard’. However, the pronunciation should also be acceptable to users of the TTS system, which obviously includes members of the general public with minimal exposure to synthetic speech. Thus, subjective evaluation is also important, particularly for proper names, which can be pronounced differently depending on the linguistic background of the speaker and other cultural factors. Because subjective evaluation is difficult compared to objective evaluation, and cannot deal with large numbers of names, there have been few attempts to do this (but see [3] who have conducted informal subjective evaluation of proper name pronunciations over the internet).

In this paper, we compare three data-driven methods for proper name pronunciation, namely: pronunciation by analogy (PbA), the table look-up method described by Weijters [4], and the table look-up method by Daelemans and van den Bosch [5]. These methods were selected on the basis that, first, we believe PbA to be the best currently-available technique for pronunciation of common words [6] and, second, table look-up is very representative of the competitor data-driven techniques to PbA. Indeed, table look-up can be seen as an alternative implementation of the broad concept of ‘analogy’. The comparison involves both objective and subjective performance.

2. Overview of the techniques

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, so converting the problem of transduction into one of classification. For the three techniques compared here, the algorithm described in [7] was used for alignment.

2.1. 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 [8], and many variants of PbA have since been based on it (e.g., [9, 10, 11, 12, 13]). The particular variant used here is now described.

When an unknown word is presented as an input to the system, so-called full pattern matching (see [12]) 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 $m$th 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 for the input 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, then the total product (TP) scoring strategy of [11] is used to select the output. The TP score is the sum of the product of the arc frequencies for all shortest paths giving the same pronunciation.

2.2. Table look-up I (TLUI)

This method was proposed by Weijters [4] who argued that his simple look-up procedure is superior to Nettalk, the well-known neural network for letter-to-phoneme conversion [14]. 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. Here, we have used $n = 7$ (heptagrams). One heptagram is produced for each letter of the input, with each letter serving in turn as the centre of the heptagram. To obtain the pronunciation for an input string, we search for the closest-fit heptagrams, i.e., those with the highest matched value between the heptagram of the input string and those of the pre-compiled look-up table. The matched values are calculated as in the pseudocode below:

```plaintext
Weight[1..7]:={ 1, 4, 16, 64, 16, 4, 1 }
MatchValue := 0
for i := 1 to 7 do
  Begin
    if windowW[i] = windowL[i] then
      MatchValue := MatchValue + Weight[i]
    end if
end for
```

Here, a heptagram in an input is referred to as $\text{windowL}$, a heptagram in a look-up table is $\text{windowW}$, and $i$ is an index into a heptagram. Weijters actually used a range of different values of $n$ and weight sets, but results did not differ too much for the different choices.

After matching, the phonemes of the closest-fit heptagrams are concatenated to form the pronunciation of the word. If closest-fit heptagrams 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.

2.3. Table look-up II (TLUII)

Daelemans and van den Bosch [5] describe a method similar to Weijters', but having defaults that could be invoked in the case of matching failure. This is expected to improve generalisation ability. In the table-construction step, all unambiguous one-to-one letter-to-phoneme mappings are searched for and stored in the 0-1-0 subtable. Then, the width of the letter window is expanded on the right by one character. All unambiguous 0-1-1 patterns are searched for and stored in the 0-1-1 subtable, excluding those patterns already in the 0-1-0 subtable. Then, the width of the letter window 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. In this work, this occurs with a 10-1-10 window. 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. Note that, 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 of the word.

3. Experimental results

In this section, we present the results obtained for objective and subjective evaluation of the three automatic pronunciation approaches as described in the previous section.

3.1. Materials

Test data were developed using a list of proper names (without pronunciations) together with the standard CMUDICT dictionary (version 0.6). The first file can be downloaded from www.festvox.org/cmu/names.lex.gz and the latter from www.speech.cs.cmu.edu/cgi-bin/cmudict. CMUDICT is a pronunciation dictionary containing approximately 100,000 word spellings and their transcriptions, includes the most frequent names and surnames in the United States (collected over 20 years ago) and their pronunciations [15]. This includes names from a wide variety of origins. The phoneme set for CMUDICT consists of 39 phoneme symbols. The procedure was simply to extract from CMUDICT pronunciations for the names on the first list. Note, however, that some names on this list were not found in CMUDICT: The number of proper
names used in this work is 52,911. The listed pronunciations were
taken to be the correct ‘gold standard’ against which to
assess the automatically-inferred pronunciations.

For TLUII, no unambiguous 0-1-0 patterns were actually
found in CMUDICT. Nonetheless, the general description of
the method in section 2.3 above remains valid.

3.2. Objective evaluation

The performances of the three methods 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 heptagrams which were
unique to a particular word and that word was stored with the
heptagram. Then, when finding a pronunciation for a particular
word, heptagrams unique to that word were removed from the
look-up table. Frequencies were left unaltered because of the
computational difficulty of adjusting them to account for the
separate removal of over 50,000 entries. (Contemporary work
has shown that ‘full’ \( k \)-fold cross validation is computationally
feasible but we nonetheless expect this simplified approach to
produce similar results.)

Results were obtained by scoring the automatically-derived
pronunciations against the pronunciations in the name list, in
terms of both words correct and phonemes correct. We have
previously argued that words correct is a more stringent mea-
sure of pronunciation accuracy and should be used in strong
preference to phonemes correct [6]. As can be seen in Ta-
ble 1, PbA achieved the highest percentage of words correct.
The difference between word accuracy for PbA and for the next
best method (TLUII) is enormously significant (binomial test,
\( z \sim 9.8, p \sim 0 \)). As expected, TLUIII achieved enormously bet-
ter performance than TLUII because of the extension to include
default tables. We note that the phonemes correct measures do
not align with words correct. TLUII achieves the best phonemes
correct score but not the best words correct score. This indicates
that phoneme errors are not independently distributed across
words, and is another reason for preferring words correct as our
measure of effectiveness. Note finally that, for simplicity, we
have ignored stress assignment in this objective evaluation.

3.3. Subjective evaluation

To assess the acceptability of the automatically-derived pronun-
ciations from the potential users’ point of view, and as a fur-
ther comparison of the three data-driven methods studied in this
work, we employ subjective evaluation by listeners of names
synthesised according to the three techniques. As a baseline,
listeners also hear pronunciations according to the CMUDICT
transcription. To keep the test times tractable, 200 names were
randomly selected from CMUDICT and pronunciations ob-
tained using each of the three data-driven methods. In the event
that all three methods gave the same (correct) pronunciation,

Table 1: Evaluation of pronunciations of 52,911 proper names
by three automatic methods in terms of words correct and
phonemes correct.

<table>
<thead>
<tr>
<th>Method</th>
<th>Words (%)</th>
<th>Phonemes (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PbA</td>
<td>65.93</td>
<td>91.03</td>
</tr>
<tr>
<td>TLUII</td>
<td>58.55</td>
<td>91.26</td>
</tr>
<tr>
<td>TLUIII</td>
<td>61.89</td>
<td>92.14</td>
</tr>
</tbody>
</table>

this was discarded and another word selected. At the end of this
process, we had 200 names whose pronunciations either dif-
ered according to the data-driven method used, or were incor-
correct (according to CMUDICT), or both. Finally, any duplicate
pronunciations were deleted to give a total of 529 different pro-
nunciations (529 \( < 200 \times 4 \)). These were then synthesised
diphone synthesis and played to listeners.

Speech output used Festival, a public domain sys-
tem intended for speech synthesis research available from
http://www.cstr.ed.ac.uk. To achieve good pronun-
ciations, it was necessary to syllabify the words and add
stress. Syllabification was done using the Festival function
lex.syllabify.phstress(PHONELIST). To obtain
stress patterns, we passed the spelling patterns through Fes-
tival’s letter-to-sound rules and then manually transferred the
stress pattern of the output to the 200 proper names. The voice
used was male KAL.

Twelve listeners, 10 male, 2 female, and all native speakers of
British English, aged between 23 and 66 years, listened to the
synthesised names over headphones in the soundproof room
in our laboratory at Southampton. We decided against online
internet-based evaluation as used by Font Litjis and Black [3]
because of the lack of experimental control this allows. Subjects
were instructed to rate their opinions on the quality of the pro-
nunciations on a five-point scale, according to the mean opinion
score (MOS) [16]. Instructions to the subjects are shown in Fig-
ure 1. Ratings were recorded by the listeners on a hardcopy list
of the name spellings, so that they could match the pronuncia-
tions heard with their spellings. The complete experiment took
some 30-35 minutes per subject.

Results are listed in Table 2. MOS values generally ex-
ceed 3.5 (i.e., tending towards ‘Almost correct’) indicating that
all methods gave reasonable pronunciations. As expected, dic-
tionary pronunciations have a higher MOS than any of the
automatically inferred methods; since the dictionary acts as the knowledge base for the data-driven methods, there is no basis on which they could systematically improve on CMUDICT.  

From best to worst in terms of MOS, the data-driven methods are ordered PbA, then TLUII, then TLUI. However, according to a paired $t$-test (assuming that scores can be treated as interval data and are normally distributed), only the difference between the dictionary pronunciations and TLUI reaches even marginal significance ($t = 1.706$, $p \approx 0.10$, df = 12 + 12 - 2 = 22). It was striking, however, that according to mean scores, all listeners but one consistently rated pronunciations in the order CMUDICT, PbA, TLUII, TLUI. The one exceptional subject reversed the order of TLUII. Thus, it seems that the data may not be normally distributed. According to a non-parametric Walsh test [17], which assumes symmetrical (but not normal) interval data, the differences between all methods are highly significant ($p < 0.005$). The Walsh test assumes related samples, which is appropriate because the listeners are the same.

4. Conclusions

We have compared three data-driven methods for proper name pronunciation: pronunciation by analogy (PbA), the table look-up method by Weijters (TLUI), and the table look-up method by Daelemans and van den Bosch (TLUII). The best result for objective evaluation was 63.93% words correct, achieved using PbA. This was enormously significantly better than the results of either table look-up method. In the latter case, TLUII was better than TLUI reflecting its superior generalisation ability. In the subjective evaluation, listeners generally rated pronunciations somewhere between ‘Acceptable’ and ‘Almost correct’ for all four conditions (dictionary pronunciation plus the three automatic methods). Although parametric paired $t$-tests did not show significant differences between methods, it was clearly evident in the MOS data that listeners consistently rated dictionary pronunciations best, followed by PbA, followed by TLUII, with TLUI rated worst. A non-parametric Walsh test showed that the differences between all methods were significant ($p < 0.005$).

In future work, we plan to test a wider range of methods (perhaps including proprietary rules if we can gain access to a set), to include automatic stress assignment, to improve PbA by using a multi-strategy approach as proposed in [12, 13] and to incorporate automatic syllabification [18].

5. References


