Semi-Automatic Generation of Pronunciation Dictionary for Proper Names: An Optimization Approach

Laxmi Narayana M
TCS Innovation Lab - Mumbai
TCS, Yantra Park, Thane (West)
Maharastra, India
laxmi.narayana@tcs.com

Sunil Kumar Kopparapu
TCS Innovation Lab - Mumbai
TCS, Yantra Park, Thane (West)
Maharastra, India
sunilkumar.kopparapu@tcs.com

Abstract

Development of a pronunciation lexicon for proper names in general and Indian proper names in particular is usually a manual effort which can not be avoided. Grapheme to phoneme (G2P) modules developed in literature are usually rule based and work best for non-proper names in a particular language. Proper names are foreign to a conventional G2P conversion module. In this paper we propose an optimization formulation for the construction of a semi-automatic pronunciation dictionary for proper names. The idea is to construct a 'small' orthogonal set of words (basis) which can span the complete set of names in a database. The transcription lexicon of all the proper names in a database can be produced by the manual transcription of only the small set of basis words. We first construct a cost function which is based on a set of features. We show that the minimization of this cost function results in an orthogonal basis. We validate the formulation with experimental results on a large real name database.

1 Introduction

Text to Speech (TTS) synthesis is an automated encoding process which converts a sequence of symbols (text) conveying linguistic information, into an acoustic waveform (speech). The two major components of a TTS synthesizer are - natural language processing (NLP) module, which produces a phonetic transcription of the given text and a digital signal processing (DSP) module, which transforms this phonetic transcription into speech (Thierry Dutoit, 1997). In general an NLP module should be able to normalize the input text and map the grapheme representation to a corresponding phonetic representation.

In general, a one to one correspondence between the orthographic representation of a word and its pronunciation is absent. In this situation, the default G2P mapping which maps a sequence of characters into a sequence of phones has to be modified according to some predefined rules. The G2P rule base is a set of rules that modify the 'default mapping' of the characters based on the 'context' in which a particular phoneme occurs. Specific contexts are matched using rules. The system triggers the rule that best fits the current context (Ramakrishnan and Laxmi Narayana, 2007). G2P converters usually produce a significant number of mistakes when converting proper names which are often of a foreign origin (Qian et al, 2006). The rule set developed is language dependent and hence an existing rule base for one language cannot be used to generate the phonetic transcription of a word from another language.

Recently Antonio et al (2007) reported that 53% is the average phonetic accuracy for proper names when a rule based methodology is used to develop a phonetic dictionary of proper names. Henk et al (2007) tried to automate the process of transcribing proper names by using a cascade of a general purpose G2P converter and a special purpose P2P (phoneme to phoneme) converter; the P2P converter learns from human expert knowledge. They

Proceedings of ICON-2008: 6th International Conference on Natural Language Processing
Macmillan Publishers, India. Also accessible from http://ltrc.iiit.ac.in/proceedings/ICON-2008

report better performance with the cascade system in comparison to using a direct rule based method. Even in the better performing cascade of G2P and P2P, more than 30% of the name transcriptions were erroneous. In a manual effort, Alen and Black (2002) adopted a web-based interface to improve pronunciation models as well as correct the pronunciations in the CMU dictionary by evaluating and collecting proper name pronunciations online.

An English G2P rule base cannot be generalized to handle all the words, especially for the proper names derived from languages like Sanskrit or other Indian languages. This means, there is a need to develop a pronunciation dictionary for proper names. But the development of such a lexicon[1] is not possible by having a mere rule set; it demands manual effort to generate phonetic transcriptions of a large set of names. A possible solution is to create a small set of words[2] which when phonetically transcribed can span and hence transcribe all the proper names in a given database. Obviously, the choice of the words should be such that they occur frequently in the database.

This paper describes a method to enable construction of the set of words[3]. We construct a cost function which when minimized results in the identification of a basis. This can then be used in phonetic transcription of the full database of proper names. The rest of the paper is organized as follows. Section 2 formulates the problem as an optimization problem with a discussion on the trivial cases of creation of a basis for proper names. Section 3 describes the proposed algorithm for basis creation of proper names. Section 4 presents the experimental results and we conclude in Section 5.

2 Problem Formulation

Given a dictionary of $|\mathcal{N}|$ proper names; two extreme and trivial cases of building a pronunciation dictionary are possible. At one extreme one could build a pronunciation dictionary by transcribing all the $|\mathcal{N}|$ names; while on the other extreme one could have a pronunciation dictionary of the 26 letters in the English alphabet and use that to construct the pronunciation dictionary of all the $|\mathcal{N}|$ names in the dictionary by concatenating the letters making the name. Obviously, the first extreme is manually intensive while the second extreme is manually easy but introduces as many joins as the number of letters in the name; as a result the pronunciation produced is pathetic. The question that one is posing is "*Is there an optimal set of words that one can identify and manually transcribe so that it can be used to produce a good pronunciation dictionary?"* In other words, is there an optimal set of words such that the need for manual transcription is *small* and at the same time the pronunciation of the names in the database is *good*? In this paper we formulate a cost function which helps us achieve a set of words (we call it the basis because it has properties of a basis) which can be used to construct the pronunciation dictionary of the full set of $|\mathcal{N}|$ names.

We make use of a restricted definition of basis (see Appendix A) to assist our problem formulation. In our case, *vector space* is the complete set of names in the database and the *basis* is a set of words such that, one can construct a name in the database by joining one or more words from the basis. Further, no word in the basis can be formed by joining one or more words in the basis. This is analogous to the scenario of concatenative speech synthesis where one looks for the longest possible speech unit to synthesize speech with minimal discontinuities.

The optimization required is that the number of entries in the basis should be as small as possible to minimize the manual effort to transcribe them and at the same time the number of basis words (joins) used to construct a name in the database should be small. These two requirements are contradicting and hence the need for optimization.

Let $\mathcal{N} = \{N_1, N_2, \ldots, N_{|\mathcal{N}|}\}$ represent all the names in the proper names database and let $\mathcal{B} = \{b_1, b_2, \ldots, b_{|\mathcal{B}|}\}$ be the basis satisfying the linear independence property of Appendix A, namely for every $b_k \in \mathcal{B}$; b_k can not be expressed as $b_i \oplus b_j \oplus \ldots \oplus b_l$[4], using any $b_i, b_j, \ldots, b_l \in \mathcal{B}$ and $b_k \neq b_i$, or b_j, or $\ldots$ b_l. This implies $b_1 \perp b_2 \perp \ldots \perp b_{|\mathcal{B}|}$. Additionally, for any name $N_p \in \mathcal{N}$, one can write

$$N_p = \oplus^n b_i \qquad (1)$$

[1] We will use lexicon and dictionary interchangeably in this paper.

[2] Words could be names themselves or part of names.

[3] We call it basis; taking cue from vector algebra.

[4] $\oplus$ Represents a join

where $b_i \in \mathcal{B}$. This is equivalent to saying that N_p can be represented by a join of some n elements in the basis set $\mathcal{B}$ which results is $\mathcal{J}_p = (n\text{-}1)$ joins. The total number of joins required to construct the entire database of $|\mathcal{N}|$ names is

$$|\mathcal{J}| = \sum_{p=1}^{|\mathcal{N}|} \mathcal{J}_p \qquad (2)$$

Observe that two trivial cases of construction of pronunciation dictionary are possible.

Case (i): If the number of joins to construct the names is to be small then all the names in the database should be present in the basis set and this would result in the largest basis, say $\mathcal{B}_{max}$ which would have all the $|\mathcal{N}|$ names in the database. Further there is a possibility that $\mathcal{B}_{max}$ is not a basis in the sense defined in Appendix A. This is shown in Fig 1.

$N_1 = b_1$
$N_2 = b_2$
$N_3 = b_3$
.
.
.
$N_{\mathcal{N}} = b_{\mathcal{N}}$

$\mathcal{B} = \mathcal{N}; |\mathcal{B}| = |\mathcal{N}|; |\mathcal{J}| = 0$

Fig 1: Trivial Case (i) where $|\mathcal{B}| = |\mathcal{N}|; |\mathcal{J}| = 0$

Case (ii): The smallest possible basis $\mathcal{B}_{min}$ would be the set of 26 alphabets in English $\{\alpha_i\}_{i=1}^{26}$ and this basis can definitely span the entire database of names $\mathcal{N}$, but the number of joins, $|\mathcal{J}|$ required to form the names in the database would be very large. This is shown in Fig 2.

$N_1 = \alpha_1 \oplus \alpha_{25} \oplus \alpha_{15} \oplus \alpha_{12} \oplus \alpha_8$
$N_2 = \alpha_3 \oplus \alpha_5 \oplus \alpha_8 \oplus \alpha_{22}$
$N_3 = \alpha_{14} \oplus \alpha_5 \oplus \alpha_{12} \oplus \alpha_2 \oplus \alpha_6 \oplus \alpha_8$
.
.
.
$N_{\mathcal{N}} = \alpha_2 \oplus \alpha_7 \oplus \alpha_{21} \oplus \alpha_{22}$

$\mathcal{B} = \{\alpha_i\}_{i=1}^{26}; |\mathcal{B}| = 26; |\mathcal{J}| \approx \infty$

Fig 2: Trivial Case (ii) where $|\mathcal{B}| = 26$ and $|\mathcal{J}| \approx \infty$

A typical plot of the number of elements in the basis $|\mathcal{B}|$ versus the total number of joins required to construct all the names in the database $|\mathcal{J}|$, is shown in Fig 3. The scenario depicted in *case (i)* corresponds to the point A in Fig 3 and the *case (ii)* corresponds to the point B in Fig 3. Probably there is a case between these two trivial solutions; like the knee point C shown in Figure 3 which can be achieved. We investigate this.

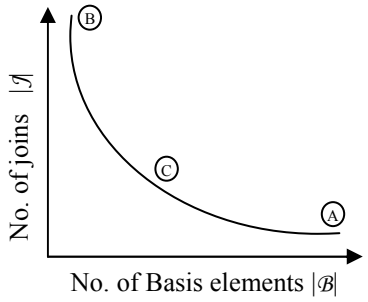

Fig 3: A plot of the number of elements in the basis, $|\mathcal{B}|$ and the total number of joins $|\mathcal{J}|$

Let $\mathcal{B}_{init}$ be an initial basis. Then a name $\{N_p\}_{p=1}^{|\mathcal{N}|}$ in the database can, in *Case (a),* be completely represented by using some of the elements in the basis, namely, $N_p = b_i \oplus b_l \oplus b_m$, where $b_i, b_l, b_m \in \mathcal{B}_{init}$ and in *Case (b)* be partially represented, namely, $N_p = b_i \oplus n_b \oplus b_m$, where $b_i, b_m \in \mathcal{B}_{init}$, and $n_b \notin \mathcal{B}_{init}$. In Case (b), for N_p to be representable using the basis, we need to necessarily add n_b to $\mathcal{B}_{init}$ and further make sure that n_b is orthogonal to all the elements in $\mathcal{B}_{init}$, namely $\{b_i\}_{i=1}^{|\mathcal{B}_{init}|}$. The addition of n_b introduces an extra element into $\mathcal{B}_{init}$, hence increasing size of the basis $|\mathcal{B}_{init}|+1$. In reality we need to keep the size of basis as small as possible. The identification of an optimal basis set reduces to an optimization problem. Specifically, to optimize a weighted sum of $|\mathcal{B}|$ and $|\mathcal{J}|$. Namely,

$$\mathcal{C} = \gamma |\mathcal{B}| + \beta |\mathcal{J}| \qquad (3)$$

Now the optimization problem can be stated as

$$\min_{\mathcal{B}} \{\mathcal{C}\}$$

Meaning, choose a basis set $\mathcal{B}$ such that the weighted sum of $|\mathcal{B}|$ and $|\mathcal{J}|$ is minimized and the sum of weights is equal to 1, namely

$$\gamma + \beta = 1 \qquad (4)$$

This formulation seeks the construction of an optimum basis that can span the entire database which can be achieved with optimal values for the two parameters $|\mathcal{B}|$ and $|\mathcal{J}|$ together. The expectation is that the optimum basis is created at some knee point C on the curve shown in Fig 3, where the number of basis elements and the number of joints are optimal.

3 The Algorithm

The pseudo code of the proposed algorithm is given in Fig 4.

```
1. 𝒩 = {N1, N2, …, Np, …, N|𝒩|}
2. initialize ℬinit = {b1, b2, …, bi, …, b|ℬinit|}
3. do{
4.   initialize ℬm = ℬinit
5.   for each name Np ∈ 𝒩 {
6.     form ℬp = {bp1, bp2, … ,bpk, … ,b|ℬp|}
          such that ℬp ⊂ ℬinit and
          bpk is a substring of Np
7.     possible_sequences(Np) with ℬp
         Np1, Np2, …,Npl, …,Npr
8.     collect all the new words
9.     for each Npl,
         obtain_cost Cpl of Npl
       (end for)
10.    choose Npl with minimum Cpl
       add new words in Npl to ℬm
11.  }(end for)
12.  isBasis(ℬm)
13.  ℬinit = makeBasis(ℬm)
14.  goto step 4
15. }until(|ℬm|-|ℬinit|)<ε[5])
16. ℬ = ℬinit
```

Fig 4: Pseudo code of the algorithm

Step 2, `initialize` $\mathcal{B}_{init}$: The basis is initialized by sorting the names in the descending order of the number of occurrences in the database and then picking up all the names whose frequency of occurrence is greater than or equal to k%[6] of the maximum frequency[7].

Step 4, `initialize` $\mathcal{B}_m = \mathcal{B}_{init}$: For reasons that will get clearer, we initialize a new set $\mathcal{B}_m$ with $\mathcal{B}_{init}$.

Step 5, `for each name` $N_p \in \mathcal{N}$, we do the following.

Step 6, `forming` $\mathcal{B}_p$: Let $\mathcal{B}_p = \{b_{p1}, b_{p2}, \ldots, b_{|\mathcal{B}p|}\}$ such that $\mathcal{B}_p \subset \mathcal{B}$ and can completely or partially construct the name N_p in the database, namely, all the basis words in $\mathcal{B}_p$ are sub-strings[8] of the name N_p. Note that $\mathcal{B}_p$ can be a null set meaning there are no elements in the basis which is a sub-string of N_p. As will be shown later, in such a case N_p should be added to the basis.

Step 7, `construction of possible sequences (`N_p`) with` $\mathcal{B}_p$: Consider $\mathcal{B}_p$ is not empty; then the words in $\mathcal{B}_p$ may partially or completely construct N_p (see Equation (1)). Let the name N_p can be constructed using the words in $\mathcal{B}_p$ in *r* different ways as shown in Fig 5.

$\mathcal{B} = \{b_1, b_2, \ldots, b_{|\mathcal{B}|}\}$

$\mathcal{B}_p = \{b_{p1}, b_{p2}, \ldots, b_{|\mathcal{B}p|}\}$

$N_{p1} = b_{p3} \oplus b_{p5} \oplus b_{p2}$

$N_{p2} = n_{p21} \oplus b_{|\mathcal{B}p|}$

.
.
.

$N_{pl} = b_{p4} \oplus n_{p31} \oplus b_{p7} \oplus n_{p32}$

.
.
.

$N_{pr} = b_{p5} \oplus b_{p6} \oplus b_{p4.}$

where

$N_{p1} = N_{p2} = \ldots N_{pl} = \ldots = N_{pr} = N_p$

Fig 5: Different ways of constructing N_p

As seen in Fig 5 there are *r* possible sequences constructed for the name N_p; The choice of the l^{th} sequence is represented as N_{pl}. For example, the sequence N_{p2} requires a new word n_{p21} which is not in $\mathcal{B}_p$, for successfully constructing $N_{p.}$ while N_{pl} requires two new words n_{p31} and n_{p32} to construct $N_{p.}$ while N_{p1} and N_{pr} completely constructs the name without the aid of any new word being added to the existing basis. If more than one such representation of N_p is possible using different combination of words in $\mathcal{B}_p$, then a decision has to be taken as to which representation is to be retained. In such case, the decision depends on the cost of constructing the sequence. That sequence which gives the minimal cost of construction is selected.

[5] ε is a small value

[6] In all our experiments we have chosen k=10.

[7] 'Maximum frequency' refers to the frequency of the name which occurs the most number of times in the database.

[8] We consider a word as a substring of a name if it is a part of the name or sometimes the name itself

Some sequences might partially construct the name. Or sometimes, none of the sequences might completely construct the name. In this case, there is a need to include some new words into the basis, to enable the basis to construct the name (and the entry should also be orthogonal to the existing basis as mentioned earlier). So, a decision has to be taken about which new word(s) should be added to the basis and what is the cost of such addition.

Step 8, `collect all the new words:` The new words required by all the sequences formed for a name are collected in a set $\mathcal{B}_m$ which is initialized with $\mathcal{B}_{init}$ (see step 4). Their frequency of occurrence is also calculated. For every name in the database, we have several possible sequences and a list of words which are not in the basis. We need to choose one of the r sequence choices to represent the name N_p. The choice is one that results in (a) minimal number of joins and (b) adds minimal number of entries to the existing basis set. Observe that there is a need for optimality in choosing one of the r sequences. We construct a cost function to identify the optimal sequence choice.

Step 9, `obtaining cost for each word sequence` N_{pl}: Let the l^{th} sequence (N_{pl}) out of the r sequences that represent N_p has n number of words, $\{s_1, s_2, ..., s_k, ..., s_n\}$ out of which η_{new} are new and η_{ex} belong to the existing basis, meaning $n = \eta_{new} + \eta_{ex}$. If η_{joins} is the number of joins in the l^{th} sequence then

$$\eta_{joins} = (n - 1) \qquad (5)$$

Let LON be the number of alphabets in the name $N_{p,,}$ then the cost function C_{pl} is given by

$$C_{pl} = \min_{1 \le l \le r} \left\{ \begin{array}{c} \dfrac{\eta_{joins}}{LON} + \dfrac{\eta_{new}}{n} \\ + max[f_{pl}\ (L_{avg}, F_{avg}, P_{avg}, SA_{avg})] \end{array} \right\}$$

The cost function is best described by looking at each element involved in the construction of C_{pl}. The number of joins in the chosen sequence need to be minimized, this is taken care by the first component in the cost function C_{pl}. The number of new words, η_{new} contributed by the chosen sequence need to be small; this is taken care by the second component of C_{pl}. If L_{avg} is the average length of the words in a sequence then,

$$L_{avg} = \left(\frac{1}{n}\right)\sum_{k=1}^{\eta} l_k$$

where l_k is the length of the k^{th} word s_k in N_{pl}. Maximization of this component reduces the number of joins in the sequence. Observe that the component $\sum_{k=1}^{\eta} l_k$ = LON. Hence, $\frac{L_{avg}}{LON} = \frac{1}{n}$. So, maximization of L_{avg} means minimization of n (number of words in the sequence) which in turn reduces the number of joins in the sequence N_{pl} (see Equation 5). With regard to the above discussion, the cost function can be rewritten as

$$C_{pl} = \min_{1 \le l \le r} \left\{ \begin{array}{c} \dfrac{\eta_{joins}}{LON} + \dfrac{\eta_{new}}{n} \\ + max[f_{pl}\ (\frac{1}{n}, F_{avg}, P_{avg}, SA_{avg})] \end{array} \right\}$$

F_{avg} is the average frequency of occurrence of the words in N_{pl} and is given by

$$F_{avg} = \left(\frac{1}{n_{new}}\right)\sum_{k=1}^{n_{new}} f_k$$

where f_k is the frequency of occurrence of s_k as a basis element, i.e., f_k is the percentage of names in the database that are in requirement of s_k for their formation, even if one of the r sequences formed for N_p requires the word. So, f_k is given by

$$f_k = \frac{\text{Number of names requiring } s_k}{|\mathcal{N}|}$$

P_{avg} is the average percentage of acceptance of the words in the sequence N_{pl} and is given by

$$P_{avg} = \left(\frac{1}{n}\right)\sum_{k=1}^{n} pd_k$$

where pd_k is the 'percentage demand' of the word from all the r sequences formed for the present name, i.e., pd_k is the percentage of r sequences formed for N_p which are demanding s_k.

$$pd_k(s_k) = \frac{\text{Number of sequences demanding } s_k}{r}$$

The binary valued attribute named by 'Syntax rule acceptance', sa_k checks if the word s_k to be

introduced follows the syntactic rules given in Appendix B.

sa_k = 1 if s_k follows the syntax rules
= 0 if it violates the syntax rules

All the entries in the basis follow the syntactic rules sa_k = 0 for the existing elements in the basis. SA_{avg} is the percentage of new words following the syntactic rules given in Appendix B. If η_{ak} is the number of words following the syntactic rules out of the η_{new} number of new words in the sequence N_{pl} (while the remaining are violating), i.e., for those words whose value of sa_k is 1, SA_{avg} can be written as

$$SA_{avg} = \left(\frac{\eta_{ak}}{\eta_{new}} \right)$$

Ideally, the factors l_k, f_k and p_{dk} should be maximum and sa_k should be 1 for any word s_k in the sequence N_{pl} for it to be optimal. This implies that the 4 parameters L_{avg}, F_{avg}, P_{avg} and SA_{avg} defined for a word sequence should be maximum. So, a function of these 4 parameters $\mathcal{f}_{pl}$ (L_{avg}, F_{avg}, P_{avg}, SA_{avg}) needs to be maximized. Different weightages can be assigned to the different parameters of this function as shown below.

$$\mathcal{f}_{pl}(L_{avg}, F_{avg}, P_{avg}, SA_{avg}) = \lambda_l L_{avg} + \lambda_f F_{avg} + \lambda_p P_{avg} + \lambda_s SA_{avg} \quad (6)$$

where λ_l, λ_f, λ_p and λ_s are the weightages assigned to L_{avg}, F_{avg}, P_{avg} and SA_{avg} respectively such that

$$\lambda_l + \lambda_f + \lambda_p + \lambda_s = 1 \quad (7)$$

Step 10, `choose` N_{pl} `with minimum` C_{pl}: One sequence among the r sequences (formed for a name N_p) which gives the minimum cost is selected and the new words, if any, present in the sequence, are stored separately ($\mathcal{B}_m$ in the pseudo code). This process is repeated for all the proper names in the database. After all the names in the database are constructed with the existing basis, we are left with a set of new words to be introduced in to the basis. We also have the frequency of occurrence of each of the new words and which database name is in requirement of a new word.

In summary, for a given name, the list of candidates from the existing basis that can construct the present name is collected and the sequences which partially or completely construct the present name are formed. Based on the cost function formulated above, one of the sequences that represent the present name is selected and new entries are made into the existing basis if required.

Step 12, `isBasis():` After adding new elements to the existing basis, there is need to check whether the basis is actually a basis or not. So, the updated basis is checked for its orthogonality, i.e., it is checked whether any of the longer words in the basis can be completely constructed with other shorter words in the basis. This task is accomplished by a function named `isBasis()`.

Step 13, `makeBasis():` If a basis element can be completely constructed with other elements in the basis, it is deleted from the basis. This task is accomplished by a function named `makeBasis()`.

Step 14, `goto step 4:` Once, an orthogonal basis is formed, the database names are again constructed with the updated basis. This completes one iteration.

In the next iteration, if some database names are not completely constructed with the pruned basis, some new entries are again made in to the basis based on the cost function formulated. The new basis is again checked for its orthogonality and pruned if necessary. The procedure of constructing the names of the database with the pruned basis is repeated again. New entries are appended to the basis if required.

Step 15, `until(|`$\mathcal{B}_m$`|-|`$\mathcal{B}_{init}$`|)<`ε`): The process of growing and pruning of the basis is stopped when no significant growth and redundancy in the basis are observed in successive iterations.

Step 16, $\mathcal{B}_m = \mathcal{B}_{init}$: The optimum basis for the generation of pronunciation dictionary for the set of proper names, is the set of words obtained in the last iteration of pruning of the basis.

4 Experimental Results

We used a proper names database[9] of 1,30,000 proper names, majority of which are Indian names. Majority of the names were made up of two parts; a first name and a second name[10]. The proper names database resulted in a set of 52897 unique

[9] Company address book

[10] We did not distinguish between a name and a surname for the set of experiments conducted. We believe that the performance of the proposed algorithm would be better if we create separate basis for names and surnames.

names. This was obtained after considering both the first and second names as two different names and removing the duplicates. So, to create a transcription dictionary one had to achieve transcription of these 52897 names. To test the performance of the proposed algorithm – we further processed the 52897 unique names. If a name consists of two or less characters then that entry as a name was not considered in the name database. This is because names with two or less characters are mostly initials of a person. To simplify computation and to understand the performance of the proposed algorithm we also did not consider the names that occurred only once in the entire database. This reduced the number of names in the database to $|\mathcal{N}| = 8157$.

The experimental results on a set of $|\mathcal{N}| = 8157$ unique names in the database are discussed in this section. These names are sorted in the descending order of frequency of occurrence in the entire database. All the names whose frequency is greater than or equal to 10% of the maximum frequency are taken as $\mathcal{B}_{init}$ which resulted in $|\mathcal{B}_{init}| = 207$ (Step 2; Fig 4). Using `isBasis()` and `makeBasis()` $\mathcal{B}_{init}$ is checked for its orthogonality (see Appendix A) i.e., the words in $\mathcal{B}_{init}$ which can be constructed from the other words in $\mathcal{B}_{init}$ are removed from it. The process of orthogonalization of basis is described in Appendix C. This resulted in $|\mathcal{B}_{init}| = 201$. The unique names in the database are then constructed by joining the words in $\mathcal{B}_{init}$. For a given name in the database N_p, the set of all names in the basis $\mathcal{B}_{init}$, which are the sub-words of N_p, $\mathcal{B}_p = \{b_{p1}, b_{p2}, \ldots, b_{pk}, \ldots, b_{|\mathcal{B}p|}\}$ is first collected (Step 6; Fig 4).

Many of the names in $\mathcal{N}$ had their sub-strings set, $\mathcal{B}_p$ empty. This is because the initial basis contains only 201 words out of which all are 'complete names'. The probability of their occurrence as a part of other names is hence very low. This resulted in many new words being appended to the initial basis $\mathcal{B}_{init}$ which resulted in $|\mathcal{B}_1| = 7133$. Using `isBasis()` and `makeBasis()` $\mathcal{B}_1$ is checked for its orthogonality (see Appendix C), and this reduced the size of $\mathcal{B}_1$ to 4545. A significant growth and reduction in the size of basis is observed in this iteration. The above process of growing and pruning (orthogonalization) of the basis is repeated for a few iterations till no significant growth or reduction in the size of basis is observed.

Recall that the weighted sum of the size of basis $|\mathcal{B}|$ and the number of joins $|\mathcal{J}|$ is to be minimized. We chose the values $\gamma = 0.8$ and $\beta = 0.2$ based on experimentation; this satisfies Equation (4). This choice of γ and β makes the basis more compact and thus reduces the manual effort required to transcribe the basis words.

Table 1 shows the weighted number of basis elements and the weighted number of joins required to construct the entire database. Fig 6 gives the variation of cost function $\mathcal{C}$ over 9 iterations.

I.No	$\gamma \lvert\mathcal{B}\rvert$	$\beta \lvert\mathcal{J}\rvert$	$\mathcal{C} = \gamma \lvert\mathcal{B}\rvert + \beta \lvert\mathcal{J}\rvert$
1	3636	968	4604
2	2812.8	1330	4142.8
3	2220	1635.2	3855.2
4	2031.2	1768.6	3799.8
5	1975.2	1817.8	3793
6	1994.4	1827	3821.4
7	1991.2	1834	3825.2
8	1996	1834	3830
9	1996.8	1834	3830.8

Table 1. $\gamma |\mathcal{B}|$ Vs $\beta |\mathcal{J}|$, $\mathcal{C}$ where $\gamma = 0.8$ and $\beta = 0.2$

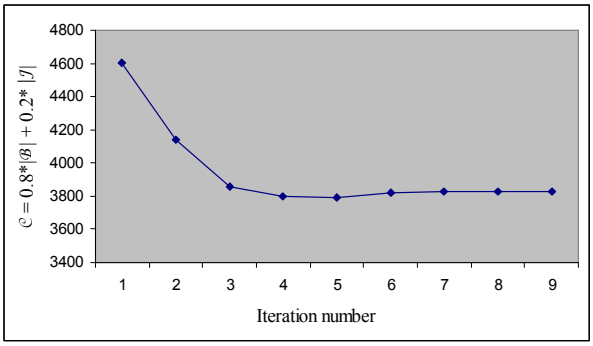

Fig 6: Variation of $\mathcal{C}$ in Equation (3) over 9 iterations $\gamma = 0.8$ and $\beta = 0.2$

As seen from Table 1, as the size of basis is decreasing, the number of joins is increasing. Over different iterations of pruning, many longer words in the basis got spaced out as they can be constructed with some short words in the basis. This result in an increase in the number of joins required to construct a name. However, with the existing basis, to construct a name in the database, longer words are given higher priority than the

shorter words or in other words, the length of a word is maximized while constructing the names in the database. For this set of experiments conducted, 'length' of the word is the only feature that is maximized in the cost function; i.e., λ_f, λ_p and λ_s are set to zero in Equation (6) while λ_l=1. So, Equation (6) is written as $f_{pl}(L_{avg}, F_{avg}, P_{avg}, SA_{avg}) = L_{avg}$ for our experimentation.

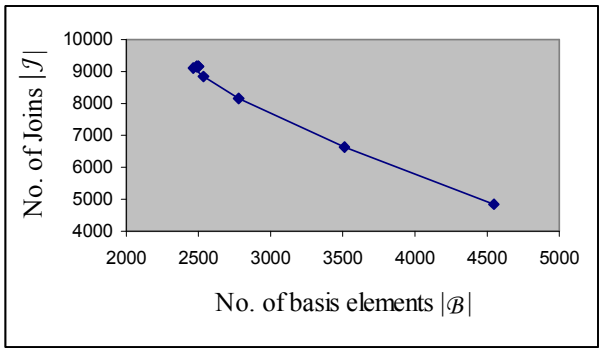

Fig 7: Graph between number of elements in the basis, $|\mathcal{B}|$ and the total number of joins $|\mathcal{J}|$

Fig 7 shows the graph between the number of joins and size of basis over 9 iterations. From Fig 7, we can see that as $|\mathcal{B}|$ decreases, $|\mathcal{J}|$ increases, and this resembles the graph in Fig 3.

5 Conclusion

Research on automatic G2P transcription has reported promising results for phonetic transcription of regular text, where G2P transcriptions follow certain rules. Generating phonetic transcriptions of proper names, where the general purpose G2P converter can not be applied directly, involves human endeavor. In this paper, an optimization approach for semi-automatic generation of pronunciation lexicon for proper names has been proposed. We first construct a cost function and the transcription problem reduces to one of minimizing the constructed cost function. Experimental results on real database of proper names show that the developed optimization framework helps in reducing the mundane task of transcribing proper names. The formulated frame work is pretty general and hence not restricted to Indian proper names. In fact, the framework is suitable for any database of proper names irrespective of language. Through experimental results we have demonstrated the working and the validity of the proposed approach.

References

A. G. Ramakrishnan, Laxmi Narayana M, *Grapheme to Phoneme Conversion for Tamil Speech Synthesis,* Proc. of Workshop in Image and Signal Processing (WISP-2007), *pp. 96-99*, IIT Guwahati, Dec 28-29 2007.

Antonio Bonafonte, Jordi Adell, Pablo D. Ag¨uero, Daniel Erro, Ignasi Esquerra, Asunci´on Moreno, Javier P´erez, Tatyana Polyakova 2007 *The UPC TTS System Description for the 2007 Blizzard Challenge,* The Blizzard Challenge 2007 -- Bonn, Germany, August 25, 2007

Ariadana Font Llitjós, Alan W Black. *Evaluation oand collection of proper name pronunciations online,* Proceedings of LREC 2002, Las Palmas, Canary Islands.

Font Llitjos, A. and Black, A. *Knowledge of Language Origin Improves Pronunciation Accuracy of Proper Names*, Eurospeech 2001, Aalborg, Denmark.

Henk van den Heuvel, Jean-Pierre Martens, Nanneke Konings 2007 *G2P conversion of names. What can we do (better)?,* INTERSPEECH 2007, August 27-31, Antwerp, Belgium.

Qian Yang, Jean-Pierre Martens, Nanneke Konings, Henk van den Heuvel, *Development of a phoneme-to-phoneme (p2p) converter to improve thegrapheme-to-phoneme (g2p) conversion of names*, lrec 2006

Thierry Dutoit, 1997. *High-quality Text-To-Speech synthesis: an overview*, Journal of Elec. and Electronics Engineering, Australia: Special Issue on Speech Recognition and Synthesis, vol. 17 no 1, pp. 25-37.

Appendix A. Basis

In linear algebra, a basis $\mathcal{B}$ of a vector space $\mathcal{V}$, by definition, is a set of linearly independent vectors that completely spans $\mathcal{V}$. $\mathcal{B} = \{b_1, \ldots, b_m\}$ is said to be a basis of vector space $\mathcal{V} = \{v_1, \ldots, v_n\}$ if $\mathcal{B}$ has the following properties:

- Linear independence property: If $a_1, \ldots, a_m$ are scalars and if $a_1 b_1 + \ldots + a_m b_m = 0$, then necessarily $a_1 = \ldots = a_m = 0$. This implies that $b_1, \ldots, b_m$ are orthogonal or $b_1 \perp b_2 \perp \ldots \perp b_m$; and
- Spanning property: For every v_k in $\mathcal{V}$ it is possible to choose scalars, $a_1, \ldots, a_n$ such that $v_k = a_1 b_1 + \ldots + a_n b_n$.

Appendix B. Syntactic rules

It is advantageous to study/analyze the words syntactically before adding them in to the basis for if the resultant basis element is not following any syntax, its phonetic representation might not properly contribute to phonetically represent a longer name which is a super set of it. Syntactic knowledge is acquired by observing the sequences formed for a name. Some rules are illustrated below.

If the word has only two alphabets, the following decisions would be taken on its candidature for the basis. V denotes a vowel and C denotes a consonant. If the word is of a particular format, the corresponding action may be taken. (Letters in bold represent the elements to be added to the basis).

- **CC – reject**
  Since the pronunciation of a phone in a sequence of phones depends on the adjacent phones, especially vowels, it is not reasonable to have a unit with only consonants. Consonants depend on the vowels for their pronunciation. So, the basis element cannot be a pure consonant sequence.
  Examples:
  - shashank – sha+sha+**nk**
  - joseph - jose + **ph**
  - shantanu – sha + **nth** + anu
  - sunny - su+**nny**
- All the words which are pure consonant strings are avoided. In other words, a basis element must have at least one vowel.
- **VC – OK**
- **CV – avoid**
- **VV – reject**
  Introducing a split between two vowels is also not reasonable, because most of the times, the combination of two vowel alphabets in English forms a dipthong. They may be two characters but their combination is a single sound.
  Example: shailendra - sh**a** + **i**lendra
- Introducing a split between sh, th, dh also should be avoided – they are two characters but their combination is a single phone/sound
  Example: bharati – bhara**t** + **h**i

Appendix C. Orthogonalization of basis

The following procedure is followed to make the basis orthogonal. Names in the basis are sorted in descending order of their lengths. For a word b_i in the basis, a set of words $\mathcal{B}_i = \{b_{i1}, b_{i2}, \ldots, b_{ik}, \ldots, b_{|\mathcal{B}i|}\}$ such that b_{ik} is a substring of b_i is collected. If $\mathcal{B}_i$ is empty, b_i is retained in the basis. For the words whose $\mathcal{B}_i$ is not empty, elements of $\mathcal{B}_i$ are sorted in descending order of their lengths. One name b_{ik} from $\mathcal{B}_i$ is considered at a time and its position is fixed in the word b_i. The remainder of b_i is filled with b_{ik} in the order they appear in $\mathcal{B}_i$. By the end of this process, b_i must have been formed completely or partially with the available elements in the basis. With one b_{ik} at a time as the first element to occupy its place in b_i, and filling the remainder of the name with $\mathcal{B}_i$, we form $|\mathcal{B}_i|$ number of sequences for b_i. If any one of the $|\mathcal{B}_i|$ sequences completely represents b_i, then b_i is deleted from $\mathcal{B}$; else it is retained.

The following example shows the sequences formed for the name b_i = 'krishna'

$\mathcal{B}_i$= krishn, krish,rish, kris, ris, ish, hna, na, kr, hn, is, ri, sh

```
krishn        Partially constructed
krish na      Fully constructed (1)
rish na       Partially constructed
kris hna      Fully constructed (2)
ris hna       Partially constructed
kr ish na     Fully constructed (3)
kris hna      Fully constructed (4)
krish na      Fully constructed (5)
kr ish na     Fully constructed (6)
kris hn       Partially constructed
kr is hna     Fully constructed (7)
ri hna        Partially constructed
kr sh na      Partially constructed
```

In the above example, the word 'krishna' in the existing basis can be constructed in 7 different ways with the other existing basis elements. So, it is not necessary to have it in the basis and hence removed from the basis.