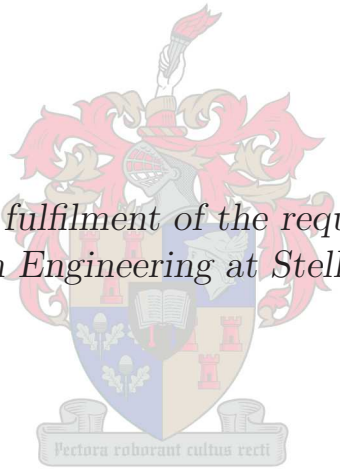
Data-Driven Augmentation of Pronunciation Dictionaries

by
Linsen Loots



Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Engineering at Stellenbosch University

Supervisor: Prof. T.R. Niesler
Department of Electrical & Electronic Engineering

March 2010

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2010

Abstract

Keywords: English accents, pronunciation dictionaries, G2P, P2P, GP2P, decision trees

This thesis investigates various data-driven techniques by which pronunciation dictionaries can be automatically augmented. First, well-established grapheme-to-phoneme (G2P) conversion techniques are evaluated for Standard South African English (SSAE), British English (RP) and American English (GenAm) by means of four appropriate dictionaries: SAEDICT, BEEP, CMUDICT and PRONLEX.

Next, the decision tree algorithm is extended to allow the conversion of pronunciations between different accents by means of phoneme-to-phoneme (P2P) and grapheme-and-phoneme-to-phoneme (GP2P) conversion. P2P conversion uses the phonemes of the source accent as input to the decision trees. GP2P conversion further incorporates the graphemes into the decision tree input. Both P2P and GP2P conversion are evaluated using the four dictionaries. It is found that, when the pronunciation is needed for a word not present in the target accent, it is substantially more accurate to modify an existing pronunciation from a different accent, than to derive it from the word's spelling using G2P conversion. When converting between accents, GP2P conversion provides a significant further increase in performance above P2P.

Finally, experiments are performed to determine how large a training dictionary is required in a target accent for G2P, P2P and GP2P conversion. It is found that GP2P conversion requires less training data than P2P and substantially less than G2P conversion. Furthermore, it is found that very little training data is needed for GP2P to perform at almost maximum accuracy. The bulk of the accuracy is achieved within the initial 500 words, and after 3000 words there is almost no further improvement.

Some specific approaches to compiling the best training set are also considered. By means of an iterative greedy algorithm an optimal ranking of words to be included in the training set is discovered. Using this set is shown to lead to substantially better GP2P performance for the same training set size in comparison with alternative approaches such as the use of phonetically rich words or random selections. A mere 25 words of training data from this optimal set already achieve an accuracy within 1% of that of the full training dictionary.

Opsomming

Sleutelwoorde: Engelse aksente, uitspraakwoordeboeke, G2P, P2P, GP2P, beslissingsbome

Hierdie tesis ondersoek verskeie data-gedrewe tegnieke waarmee uitspraakwoordeboeke outomaties aangevul kan word. Eerstens word gevestigde grafeem-na-foneem (G2P) omskakelingstegnieke geëvalueer vir Standaard Suid-Afrikaanse Engels (SSAE), Britse Engels (RP) en Amerikaanse Engels (GenAm) deur middel van vier geskikte woordeboeke: SAEDICT, BEEP, CMUDICT en PRONLEX.

Voorts word die beslissingsboomalgoritme uitgebrei om die omskakeling van uitsprake tussen verskillende aksente moontlik te maak, deur middel van foneem-na-foneem (P2P) en grafeem-en-foneem-na-foneem (GP2P) omskakeling. P2P omskakeling gebruik die foneme van die bronaksent as inset vir die beslissingsbome. GP2P omskakeling inkorporeer verder die grafeme by die inset. Beide P2P en GP2P omskakeling word evalueer deur middel van die vier woordeboeke. Daar word bevind dat wanneer die uitspraak benodig word vir 'n woord wat nie in die teikenaksent teenwoordig is nie, dit bepaald meer akkuraat is om 'n bestaande uitspraak van 'n ander aksent aan te pas, as om dit af te lei vanuit die woord se spelling met G2P omskakeling. Wanneer daar tussen aksente omgeskakel word, gee GP2P omskakeling 'n verdere beduidende verbetering in akkuraatheid bo P2P.

Laastens word eksperimente uitgevoer om die grootte te bepaal van die afrigtingswoordeboek wat benodig word in 'n teikenaksent vir G2P, P2P en GP2P omskakeling. Daar word bevind dat GP2P omskakeling minder afrigtingsdata as P2P en substansieel minder as G2P benodig. Verder word dit bevind dat baie min afrigtingsdata benodig word vir GP2P om teen bykans maksimum akkuraatheid te funksioneer. Die oorwig van die akkuraatheid word binne die eerste 500 woorde bereik, en ná 3000 woorde is daar amper geen verdere verbetering nie.

'n Aantal spesifieke benaderings word ook oorweeg om die beste afrigtingstel saam te stel. Deur middel van 'n iteratiewe, gulsige algoritme word 'n optimale rangskikking van woorde bepaal vir insluiting by die afrigtingstel. Daar word getoon dat deur hierdie stel te gebruik, substansieel beter GP2P gedrag verkry word vir dieselfde grootte afrigtingstel in vergelyking met alternatiewe benaderings soos die gebruik van foneties-ryke woorde of lukrake seleksies. 'n Skamele 25 woorde uit hierdie optimale stel gee reeds 'n akkuraatheid binne 1% van dié van die volle afrigtingswoordeboek.

Acknowledgements

Thank you to:

- Prof. Thomas Niesler, for the best supervision I could ask for. He is not only intelligent and experienced, but also goes to great lengths to provide his students with all the aid and advice he can.
- My parents, for their support and wisdom during my studies.
- Alison Wileman, for transcribing SAEDICT, as well as her assistance with and interest in my use of this dictionary.
- Dr. Gert-Jan van Rooyen, for this thesis template.
- The Wilhelm Frank Bursary for a substantial bursary in 2009, as well as the NRF[1] for financial assistance in 2008.
- Everyone in the DSP lab, especially Pieter Müller and Pieter Oosthuizen, for their friendship over the past two and a half years.

[1] National Research Foundation

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

ASR	automatic speech recognition
DP	dynamic programming
DTW	dynamic time warping
EM	expectation-maximisation
G2P	grapheme-to-phoneme
GenAm	general American accent
GP2P	grapheme-and-phoneme-to-phoneme
HMM	hidden Markov model
LTS	letter-to-sound
P2P	phoneme-to-phoneme
PbA	pronunciation by analogy
POI	probability of improvement
RP	received pronunciation
SAE	South African English
SSAE	Standard South African English
TTS	text-to-speech

Variables

symbol	**description**
$H(\cdot)$	entropy
$i(t)$	the entropy of a node t
Δi	entropy gain

Chapter 1

Introduction

Speech is the primary way in which we interact with people and even the way in which we frame our thoughts. As society places increasingly more focus on the use of computers and electronics, it is desirable to make this interaction easy and comfortable for users. The best way to accomplish this is to build computers able to speak and understand our language. This is particularly useful within the South African context, where many people do not have the training to operate computers normally and are not fluent in the language, usually English, in which the computer functions.

For speech technology to allow computers to successfully communicate with humans using speech, they must be able to both create and understand speech. As a result, both speech synthesis and speech recognition are extremely important to developing a functioning speech interface.

1.1 Speech synthesis

Speech synthesis is, broadly, the generation of speech by a machine. This was initially done by mechanical-acoustic means. In recent times, however, it has become almost exclusively the domain of computers and electronics.

The applications of speech synthesis are manifold. Possibly the most widely used application at present is in automated dialogue systems such as those found in customer call centres. Other common uses include text to speech (TTS) systems for the disabled, particularly the blind, and educational systems such as those that help people learn new languages.

Computer speech synthesis falls into two broad categories: concatenative synthesis and rule-based synthesis. Concatenative synthesis involves the joining and blending of many pre-recorded sections of speech to form new utterances. Rule-based synthesis is the generation of speech signals purely from rules, usually using parameters determined by analysing recordings of natural speech.

1.1.1 Text preprocessing

A complete TTS system contains a good deal more than an audio synthesiser able to generate intelligible speech sounds. The rest of the system can be broadly categorised as text preprocessing. This includes all the analysis of the given text (lexical, syntactic and semantic) needed to determine the phonetic properties, as well as the prosody (the pitch and duration of those sounds), of the speech to be generated. Text preprocessing requires a number of steps, which are broadly identified by Reichel and Pfitzinger as the following [35]:

Text normalisation Text normalisation consists of three separate processes, namely sentence *segmentation*, which entails identifying sentence boundaries, *tokenising*, or identifying individual words, and converting *non-standard words*, especially abbreviations and numbers, to their spoken equivalent.

Part of speech tagging This step entails determining different words' grammatical or syntactic categories. This is necessary both for pronunciation, which is at times determined by part of speech, and for stress assignment, where part of speech and sentence structure play a major role.

G2P conversion Grapheme-to-phoneme (G2P) conversion is perhaps the most significant step in text preprocessing, and involves the transliteration of words from orthographic (written) form to a phonemic transcription. This is especially challenging for languages such as English, where the correspondence between letters and sounds is often ambiguous. A related step is the determination of the duration and pitch of the phonemes.

Word stress Once the pronunciations of words have been determined, stress needs to be assigned to syllables and words. This needs to take into account morphological information about individual words, as well as syntactic information about entire sentences or even passages of text.

While all the aspects of text preprocessing are important, the most necessary step for a TTS system to produce intelligible speech is G2P conversion. A lot of work has been done in this regard, using both manually-created rules and automatic, data-driven techniques. Excluding rules hand-crafted by linguistic experts, all G2P conversion is reliant on a pronunciation dictionary, either as a direct source of pronunciations or as a source of training data from which rules can be automatically created.

1.2 Speech recognition

Automatic speech recognition (ASR) is the development of systems that are able to take audio speech signals as input, analyse them and automatically produce textual transcripts. Sufficiently accurate speech recognition is thus a prerequisite for using speech as input for any electronic system or computer.

Recognition commonly involves a number of different steps. First acoustic signals are analysed and feature vectors are extracted from them. These vectors can then be classified as a certain phoneme or phonemes - usually a number of potential phonemes, each with an associated probability. Lastly the phonemes thus determined are combined and converted into possible words. Complex language models that store information about likely sequences of words are used to determine which of the possible word sequences is most likely.

For accurate ASR it is thus essential that strings of phonemes can be converted into words. In order to do this, it is essential to have a pronunciation dictionary that contains words' orthographic and phonemic transcriptions - such a dictionary can then be queried with the phonemes produced by the acoustic analysis of the speech, and the most likely word or words matching those phonemes found.

1.3 Pronunciation dictionaries

It is clear from the preceding sections that both TTS and ASR are reliant on pronunciation dictionaries, either to convert words into phonemic transcriptions for synthesising, or to convert phonemes back into written words. G2P conversion allows the automatic generation of pronunciations, and thus the supplementation of pronunciation dictionaries. Furthermore, most G2P conversion techniques can be effectively applied to the reverse problem, phoneme-to-grapheme conversion [29, 36]. We have focussed on G2P conversion in this work.

The simplest method of performing G2P conversion is by means of a direct lookup in a large pronunciation dictionary. This is computationally fast (assuming an efficient lookup scheme is used), and should be error-free. There are significant shortcomings, however. The most important is the unavoidable occurrence of out-of-vocabulary words - there must be some systematic way to determine pronunciations for new, unseen words. Another disadvantage is that the lexicon typically has to be created by hand, which is slow, expensive and (especially where multiple annotators are involved) sometimes inconsistent.

In addition to these problems, pronunciation dictionaries can also require a lot of storage space. By exploiting correspondences between orthography and pronunciation, the memory requirements of a G2P converter can be considerably less than that of a full pronunciation dictionary [8]. In this way, G2P converters are also used to achieve lexicon compression [32].

1.3.1 Out-of-vocabulary words

No dictionary can ever contain all the possible words that may need to be pronounced. While the sheer number of words that exist in most languages already present considerable difficulty, it is the dynamic nature of any real-world language that makes complete coverage impossible. Neologisms (new words) are created all the time, and a dictionary would have to be constantly updated to remain all-inclusive. This ongoing augmentation of a pronunciation dictionary is usually practically unfeasable, since it requires constant human intervention.

Furthermore, considerable time and expense are associated with the manual maintenance of a pronunciation dictionary by human language experts. For this reason it is desirable to automate as much of the process as possible.

Humans, when presented with an unknown word, are usually able to pronounce it correctly. There must therefore be a sufficiently strong underlying relationship between the spelling and the pronunciation that can be used to train a machine to perform the same conversion. For this reason it seems probable that a G2P system should be able to convert words it has never encountered before.

Most systems will use a combination of a dictionary and such a system, automatically generating pronunciations only when the word in question is not present in the dictionary [15]. Alternately, systems can use an exception dictionary, explicitly storing pronunciations only for those words whose pronunciation the system is known to generate incorrectly [28].

1.3.2 Storage space

A pronunciation dictionary commonly contains tens of thousands of words, with large dictionaries easily containing upwards of 100 000 entries. Storing this many words and their pronunciations can require a significant amount of memory. While recent increases in available storage have to a large extent alleviated this problem, it still plays a role, especially in handheld or embedded devices with limited resources.

An automatic G2P conversion system relies on some set of rules or correspondences, either explicit or implicit, by which grapheme strings can be converted to phoneme strings. Beyond providing pronunciations for unknown words, such systems can exploit patterns in the pronunciation dictionary, which can result in significant compression with regard to storage space. A simple method of implementing such data compression without any loss of information entails storing the minimum amount of context to uniquely determine a given grapheme's corresponding phoneme [45]. This is done by generating rules with progressively wider context windows, only creating a rule if it is able to uniquely specify a grapheme-phoneme match, until all words in the dictionary are covered. This approach yielded 94.2% compression when applied to Dutch [45], a language with a fairly regular letter-to-sound (LTS) correspondence.

1.3.3 Exception dictionary

It is common practice for a G2P system to make use of an exception dictionary to store words that are known to be transliterated incorrectly by the automatic process. The size of such a dictionary will of course depend on the accuracy of the system - a better converter (which usually implies one requiring more storage) will make fewer errors, and require a smaller exception dictionary.

An effective approach to compressing this exception dictionary is to store only corrections

to the (incorrect) phoneme string generated by the system, rather than storing the entire pronunciation of the word [28]. This takes advantage of the fact that even words for which incorrect phoneme strings are generated seldom contain more than one or two incorrect phonemes. These individual errors can then be marked as *index-value* pairs, with the index indicating which phoneme is incorrect and the value giving the correct phoneme. By applying these corrections, the correct pronunciations can be generated. For any word in which most of the phonemes are correct, this will lower the storage needed for the correction, resulting in a considerable overall decrease in storage space required.

1.4 Symbol alignment

Most approaches to performing G2P conversion are based on a *classification* method: the graphemes are considered to be observations, and the corresponding phonemes are different classes to which the graphemes must be assigned. For this to be possible, there must exist a one-to-one correspondence between the graphemes and the phonemes. In practice, however, the relationship between graphemes and phonemes is many-to-many. In order to use standard classification techniques, it is thus necessary to create an alignment between the graphemes and the phonemes. While it is possible to align graphemes and phonemes by hand, this is a highly labour-intensive process and one which can be automated quite successfully.

A grapheme-phoneme alignment reduces the complexity of the G2P conversion problem by allowing each grapheme, with its context, to be considered separately by the system, and a single phoneme, or class, to be produced.

For a completely intuitive alignment, there would exist a many-to-many relationship between the graphemes and phonemes. This would necessitate inserting nulls or grouping symbols in both the graphemic and phonemic strings. It is important to note, however, that modifications (the insertion of nulls and the grouping of phonemes) can only be made to the phoneme string. This is because, when the system is used for G2P conversion, only the raw grapheme string is available. Because the system's eventual input contains a grapheme string with no nulls or grouped graphemes, it is best to keep the training data in the same format.

Graphemes	e	x	t	r	e	m	e
Phonemes	*eh*	*k s*	*t*	*r*	*iy*	*m*	-

Table 1.1: *An example of grapheme-phoneme alignment for the word "extreme".*

Table 1.1 shows an example of alignment: the word "extreme" after its phonemes and graphemes are aligned. While most of the graphemes in the word correspond only to a single phoneme, it contains both an insertion (the 'x' corresponds to two phonemes, /*k s*/),

and a deletion (the final 'e' has no corresponding phoneme). The problem of insertions and deletions, along with possible solutions, is discussed in Section 2.1.

1.5 Data-driven G2P conversion

This section will discuss various data-driven techniques that have been used to perform G2P conversion.

The first G2P conversion systems made use of hand-written rules created by expert linguists [15]. This is an expensive process, requiring expert knowledge as well as careful testing and maintenance. Furthermore, when developing TTS systems for smaller languages, it is often not a viable option as both funding and experts are in short supply. As a result, data-driven, automatic learning approaches have received considerable attention, and are able to provide a good alternative. A number of distinct methods have been employed to perform G2P conversion. The most commonly used techniques are [50]:

- Decision trees
- Bayesian (stochastic) techniques, including HMMs
- Pronunciation by analogy (PbA)
- Neural networks

These four alternatives are reviewed in the following sections.

1.5.1 Decision trees

A very popular technique for automated G2P conversion is the use of decision trees. Decision trees consist of directed acyclic graphs, beginning at a single root node. This root node has a number of children, and each of those has children, and so forth. Each branch node (i.e. each node that is not a leaf, and thus has child nodes) contains an attribute giving information about the type of instances that may belong to it. Such attributes are usually phrased as questions relating to the context of a specific instance, for example *"is the grapheme 2 positions to the right an 'a'?"*. Finally, all nodes are associated with a class, or in the case of G2P conversion, a phoneme [39]. An example of such a tree is shown in Figure 1.1.

When using a decision tree to classify a grapheme, one starts at the root of the tree. The question associated with this node is asked, and if there is a child node corresponding to the answer, one moves to that child node. This process is repeated until no further movement is possible (usually at a leaf node), in which case the class of the current node is taken as the grapheme's class.

While decision trees need not be binary, most implementations use binary trees. This is advantageous primarily because it enables questions to be phrased in a true/false form, and also means that any traversal must end at a leaf.

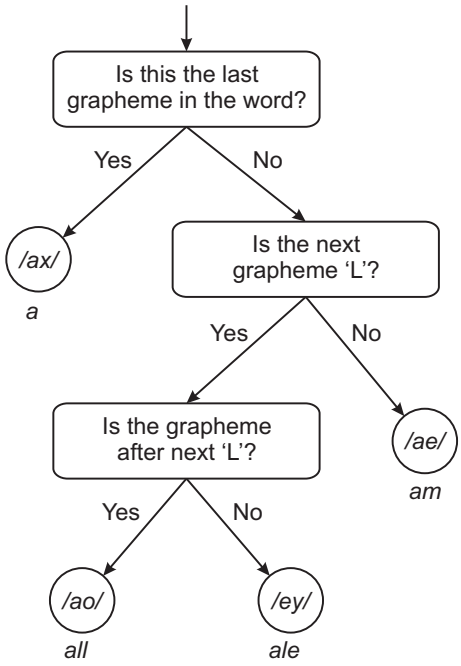


Figure 1.1: *Sample decision tree able to determine the first phoneme of* a*,* am*,* ale *and* all*.*

Tries

A trie (short for reTRIEval) is another tree-based data structure commonly used to store information like that found in a dictionary. Each node in the trie, excluding the root node, contains a single letter. Words are represented by paths from the root to a node in the tree - the concatenation of the letters of all the nodes in the path (taken in order) form the word. Nodes are given markers to indicate whether a word terminates at the node or not, and a word's pronunciation is stored at the node where it terminates. This is necessary because some words are entirely contained within other words (*run* is a substring of *running*), and as a result do not terminate at leaf nodes. Tries form an efficient way to store a lexicon, and allow lookup operations to be performed very quickly. An example of a trie encoding a lexicon is shown in Figure 1.2.

Tries have been suggested by some sources as a method for performing G2P conversion [2]. For this application, however, individual phonemes are stored rather than entire words. Each phoneme's context information forms the string by which it is placed in the trie. This context is usually the phoneme's corresponding grapheme, with just enough of its closest adjacent graphemes to ensure that the context can specify only the phoneme in question.

This context is then used to store the phoneme in the trie, much as words' orthography was used to store them in the example in Figure 1.2. The only important difference, however, is that these context strings are read from the focus grapheme outwards, rather than from left to right. This allows only the minimum necessary context to be used for each phoneme.

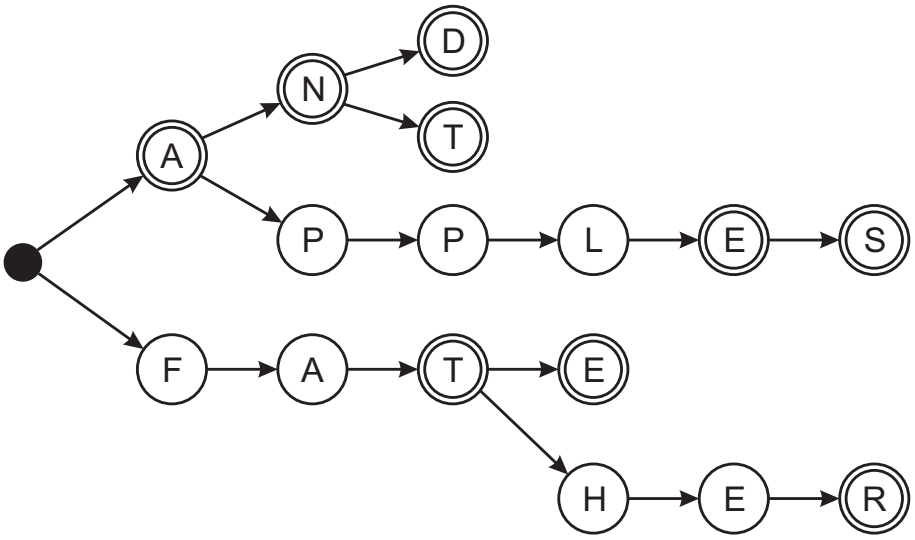


Figure 1.2: *Trie encoding the words* a, an, and, ant, apple, apples, fat, fate *and* father. *Nodes with double lines indicate word termination points.*

By storing all the different context patterns present in a dictionary, a trie can in this way store the entire dictionary without any loss of information. If queried with an unknown set of graphemes, the closest match found in the trie is used.

Tries are efficient structures for compressing a pronunciation lexicon. Due to their tree structure, however, tries cannot generalise any better than a decision tree - when queried with an unknown word they will merely function as a type of decision tree.

1.5.2 Stochastic models

An alternative method of performing G2P conversion is to use statistical techniques to develop a probabilistic model of the grapheme-phoneme relationship, and then analyse the model to determine the phoneme string most likely to correspond to a given grapheme string. A number of different approaches have been used, of which the most important are Hidden Markov Models (HMMs), and joint n-grams.

Hidden Markov models

An HMM is a type of probabilistic graphical model. It is a network with edges representing probabilities and connecting nodes, which represent different states. The distinguishing characteristic of an HMM is that the states are hidden, and can only be indirectly observed through the output symbols generated by these states. These output symbols or values are themselves produced according to some state-specific probability distribution. The model is thus doubly stochastic, as both state transitions and output symbol generation are random processes.

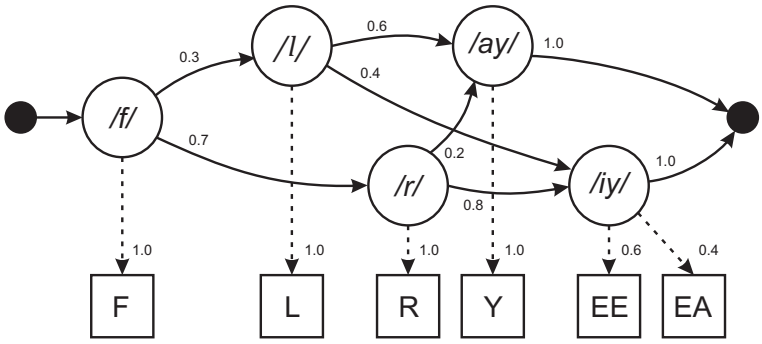


Figure 1.3: *HMM able to generate graphemes for the words* fry*,* free*,* fly*,* flee *and* flea*.*

HMMs are commonly used in other fields, such as speech recognition, to classify a sequence of observations where there is uncertainty both about the accuracy of the observations and the hidden state sequence they represent. They are particularly successful for linear sequences, as are often found in speech processing.

The application of HMMs to G2P conversion, proposed by Taylor [40], is thus not an unintuitive one. The models are set up so that the hidden states represent the phonemes, and the graphemes are regarded as the noisy output from those states. An HMM is then used to find the most likely hidden state sequence, i.e. phoneme string, given the output sequence, i.e. graphemes. By representing each phoneme's output as a set of possible graphemic subsequences, each relating to a group of graphemes that can correspond to that phoneme, this process can be simplified. An example of such an HMM is shown in Figure 1.3.

A further advantage of HMMs is that if a set of phonemic constraints (i.e. rules specifying which phonemes can follow one another and so forth) are known, that knowledge can be incorporated into the graph to ensure the generated pronunciation is at least phonemically possible [40].

This approach has some limitations, however. The most notable is the lack of graphemic dependencies. While the use of higher-order models can increase the context of the model by making probabilities dependent on several prior states (rather than only the single previous state), these dependencies are still restricted to the previous phonemes. Any dependencies between adjacent graphemes, or between a phoneme and the graphemes corresponding to a different phoneme, cannot be directly captured by an HMM. This is because the graphemes, as output symbols, are dependent only on the current state. They are thus independent of both other graphemes observed for the same state, and of previous states [17, 40].

The results in the literature indicate that without extensive preprocessing and knowledge-based techniques, HMMs are unable to match the performance of decision trees [40].

Joint models

Various authors have proposed the use of a joint model, whereby states are based on a grapheme-phoneme pair, or a *graphone* [19]. An n-gram model is then constructed around these graphones, and used to predict the most likely state sequence (from which the phoneme string can be determined) given a string of graphemes. The use of graphones addresses the major limitation of HMMs, namely that graphemic relationships are not modelled directly.

As with most methods, an alignment between graphemes and phonemes in the training data is required. Unlike decision trees, however, many-to-many relationships can quite easily be handled by allowing many-to-many graphone mappings. Most authors therefore consider a graphone to contain a grapheme string and a phoneme string, each of length zero or more [19].

A joint n-gram model functions in a similar manner to an HMM, with the exception that the output of any given state (i.e. the graphemic part of its graphone) is known with certainty. Demberg et al [17] even used an HMM with unity output probabilities to implement such an n-gram model. A complete joint n-gram system is described in some detail in [8]. The model is trained using the EM algorithm, and G2P conversion done by searching for the most likely path that corresponds to the given grapheme string.

As the history (n) of an n-gram model increases, the data sparseness will increase correspondingly. As a result it is important to use effective smoothing algorithms so that the model still performs well in cases for which no training data existed [12]. The number of states also increases exponentially with n. This limits the context that can be used in a practically realisable system. This leads to another shortcoming of n-gram models, one that occurs in almost all G2P techniques, namely their inability to model long-range dependencies in a word's pronunciation [18]. An example of this is the change in pronunciation at the beginning of the word caused by changing *photograph* to *photography* - a graphemic change only at the end of the word. A possible method to at least decrease the effect of this problem is to use *x-grams*, variable-length n-grams. These can considerably decrease the size of the model without any notable effect on performance [33], or alternatively allow greater history to be taken into account without significantly increasing the size of the model.

According to Huang et al [19], graphone-based techniques achieved accuracies comparable to those of decision trees. They report phoneme and word accuracies of 87.8% and 44.7% respectively, with a comparable decision tree achieving 88.4% and 50.1% respectively.

1.5.3 Pronunciation by analogy

PbA is an approach to G2P conversion that aims to implicitly model relationships between grapheme and phoneme strings. This is done by repeatedly taking substrings of the word to be converted, matching those substrings to words in a training dictionary to predict a pronunciation for each substring, and then concatenating all these partial pronunciations to form the final pronunciation [27]. This approach requires a large amount of storage space

(to hold the lexicon), and a fair amount of processing power, because queried words must be compared to all potential matches in the dictionary.

There are a number of different ways to perform both the string matching and the final selection of a phoneme from those present in the matched strings [27]. One approach is to create pseudo-morphemes, essentially commonly-occurring substrings, from the training dictionary, and store their respective pronunciations. These pseudo-morphemes are then used to find the phonemes corresponding to new words, by attempting to match parts of the word to known pseudo-morphemes [43]. Since the pseudo-morpheme can be found in advance, it should operate faster at run-time. This comes at a price, however: it is possible that there will be sequences in the training data that are not stored as pseudo-morphemes, and which can thus not be matched to new words.

Bellagarda [6] takes a more complex approach in which all words bearing a similarity to the out-of-vocabulary word are found using their *orthographic neighbourhood*, a measure of how close the the words are to one another. From these words substrings are constructed, and then matched to the unknown word to generate a phoneme string. This method is intended to work well with words such as proper nouns, which often don't follow the standard rules or are borrowed from other languages and have resultantly different pronunciations.

While the results produced by PbA are comparable to other techniques, this approach requires more complex algorithms and has not yet been sufficiently developed to provide a better alternative [6].

1.5.4 Neural networks

A final technique that has been applied to G2P conversion, albeit not widely, is the use of neural networks. The best known example is the NETTalk system [38], which uses a multi-layer perceptron trained using standard back-propagation [50]. The network has an input layer, a single hidden layer and an output layer. These layers operate on a purely feed-forward manner. The input to the network is, as with other approaches, a grapheme and a context window of surrounding graphemes. Previously derived phonemes can also be used as part of this context.

An alternative approach, suggested by Adamson and Damper [1], is to use a recurrent network. This network is temporal, in the sense that sentences are viewed as dynamic sequences of words, rather than static graphemes.

Neural networks provide models that are considerably smaller than those based on other techniques like decision trees [21]. The use of a connectionist architecture for G2P conversion is limited, however, and cannot match the performance levels of other methods. They also do not seem to handle large training sets well, and generalise poorly to irregular words such as proper nouns [50].

1.6 General considerations

There are a number of concerns that are common to all G2P conversion systems, as well as other factors that can influence or improve any such system. These factors are discussed below.

1.6.1 Syntax

The pronunciation of words in many languages is not exclusively determined by their spelling, but depends also on the syntax of the sentence being considered. An example is the English word *read*, which is pronounced differently depending on its tense (past or present). If a complete text preprocessor is to be constructed, words will need to be assigned a part of speech, as well as other syntactic information. This information can then be included in the context used for G2P conversion. As syntactic analysis is a complex problem and words for which the pronunciation is ambiguous are relatively few, the subject will not be further investigated here.

1.6.2 Morphology

Morphology, or the study of how words are formed from meaningful roots and affixes, is something that humans make considerable use of when reading aloud. It is also a topic that can provide useful information when automatically performing G2P conversions. Morphological analysis, however, traditionally needs expert knowledge and is as a result largely language-specific.

It is possible to automatically construct morphological analysis rules, but this requires a training dictionary that contains morphological decompositions, something that is not always readily available. Any attempt to create such decompositions purely from the spelling of words becomes merely a form of PbA, and cannot discover true morphological relationships. As a result, an attempt to discover such decompositions and then use them as part of the G2P conversion provides no improvement on a direct conversion [17]. Should a morphological dictionary be available, other G2P techniques can be developed to exploit the additional information. Due to the unavailability of such a dictionary in the South African context, these techniques have not been investigated further.

1.6.3 Syllabification & stress assignment

The correct pronunciation of a word requires not only the phonemic transcription, but also the syllable boundaries and stress placement. While sentence-level stress assignment requires natural language processing (both syntactic and semantic), the intra-word stress can be learned from a correctly annotated pronunciation dictionary. Demberg et al [17] report that noticeable improvements can be obtained by including stress assignment, as well as

syllabification, in the G2P conversion process. This can easily be done by adding stress and end-of-syllable output markers to the system.

1.6.4 Context

Because there is not a direct, one-to-one correspondence between graphemes and phonemes, all approaches to G2P conversion must rely on the context within which a grapheme occurs to decide what the corresponding phoneme in the pronunciation must be. The choice of what constitutes this context is thus a significant factor in the success of the total system. The most important context is undoubtedly the grapheme itself (often termed the focus grapheme), followed by its neighbouring graphemes. Other attributes that can form part of the context include previously generated phonemes (assuming a linear system that operates sequentially from the first to the last grapheme of the word) and the location of word boundaries.

Webster and Braunschweiler did a comprehensive analysis of potential features with which to train a G2P conversion system [48]. The only features which yielded an improvement above those already mentioned were stress (using a morphological stress predictor), and a window of the vowels surrounding the focus grapheme (as an approximation to the surrounding syllables). As no South African lexicon containing stress is currently available, this was not a viable option for this project. The addition of vowel letters was not reported to have a very large impact, but is a possible improvement.

Window size

The number of neighbouring graphemes that are included in the context needs to be considered first. Most systems take a window of fixed size, and disregard any graphemes outside this window, as the information gained from neighbouring graphemes has been found to decrease with their distance from the focus grapheme [45]. There are words, however, for which a very large context window is needed to accurately determine their pronunciation, such as the previously mentioned example of *photograph* and *photography*. There is a trade-off, however, since a large context window results in very sparse data sets. This, in turn, can cause the system to become over-trained, and unable to make good generalisations when presented with unknown words. Torkkola found that, for English, the increase in performance decreases rapidly when considering a context width of more than four letters (e.g. two to the left and two to the right), and that this is thus a fairly optimal window size [42].

Directionality

When choosing a context window, it is not necessary to choose one symmetrical around the focus letter. For English it was reported that letters to the right of the focus letter are more important than those to the left, and that a good balance is obtained taking 1/3 of the window before the focus letter and 2/3 after [42]. A possible reason for this is the fact that

suffixes give information regarding the morphology and pronunciation of an English word more often than prefixes do [32].

If phonemes that have already been derived are included in the context by which the next phoneme is determined, it becomes a concern whether words are converted from the left or from the right. Including phonemes can give an improvement in accuracy, and when they are included the generated phonemes are, according to most accounts, more accurate if the conversion takes place from right to left [32].

1.6.5 Input and output coding

The manner in which data is encoded, both regarding input to the classifier (the focus grapheme and its context) and the output of the classifier (the phoneme relating to the grapheme in question) can have an impact on the performance of the system.

Most systems encountered encoded both input and output symbols as single, enumerated characters. The only alternative method encountered was to represent features in a binary format, as discussed by Bakiri and Dietterich [3]. In this approach input symbols are first each encoded as 29-bit binary strings, with each bit representing a separate letter (the additional three bits represent a comma, period and blank). Each letter thus has only a single 1 bit in the string, with the rest all 0. All the different letters' strings are then concatenated to make a single, long binary string to analyse, rather than a number of shorter strings. This method was specifically designed to be used with neural networks and has received no attention in other studies.

The other aspect of this binary coding method relates to the encoding of the output symbol (the phoneme). Here binary coding was also employed, but with different bits representing different features of the phoneme, such as voicing (voiced / unvoiced) or place of articulation (labial, velar, glottal etc.). The G2P conversion function could then be broken up into separate functions, each relating to a single bit in the output binary string. Once all the bits are generated, the resulting string can be mapped to the phoneme with the string most closely matching the output string. This has the advantage that error-correcting output codes can be used so as to better separate different phonemes, compensating for some conversion errors. This comes at a price, however: the binary functions are learning a feature with no direct relationship to a real-world property, and one which may thus be more difficult to learn accurately. The result is that the memory footprint of the system is somewhat larger [3].

1.7 Project scope and contribution

This project aims to investigate ways in which a pronunciation dictionary can be augmented using automatic, data-driven algorithms.

The first method discussed will be G2P conversion using decision trees. Here the workings of decision trees will be discussed in detail, as well as the different ways in which they can be adjusted to improve their performance.

Next two novel approaches are suggested by which the same decision tree techniques are used to convert pronunciations from one accent to another. These methods will be termed phoneme-to-phoneme (P2P) and grapheme-and-phoneme-to-phoneme (GP2P) conversion, in acknowledgement of their G2P roots. They are particularly useful for less prominent accents, like South African English (SAE), for which it is expensive and time-consuming to develop a new dictionary. Instead, the availability of extensive dictionaries for British and American English can be taken advantage of to obtain SAE pronunciations. It would thus be advantageous if these could easily be converted to SAE. This work has been published in a paper presented in Brighton, UK, at Interspeech 2009 [26], and has subsequently been submitted as an article to the journal Speech Communication. It also forms the base of a paper presented at PRASA 2009 which compares the developed approach to one proposed by the Meraka Institute [25].

Finally, a study is carried out into the number of words required by the G2P, P2P and GP2P algorithms to achieve accurate pronunciations of new words in a pronunciation dictionary. This study sheds light on the comparative efficiency of the three algorithms. In addition, it also identifies the sequence of words required in an SAE dictionary for these algorithms to be most effective.

1.8 Thesis overview

The structure of this thesis will broadly follow the project scope set out above. The next chapter will provide a detailed study of decision trees for G2P conversion. It will describe how they are created and optimised, and what the optimal parameter configuration is. Experimental results are given to show how accurately they perform G2P conversion for American, British and South African accents of English.

Chapter 3 will start with a phonetic comparison of these three English accents. Then the application of decision trees to the conversion of pronunciations between these accents, by means of both P2P and GP2P conversion, is presented in detail, followed by experimental results. In Chapter 4 the study of the number of words needed for accurate G2P and inter-accent conversion is described. Both the methodology as well as experimental results are presented. Chapter 5 provides an overview of the system implementation, and the thesis concludes with Chapter 6, which gives a summary of the overall project, provides some conclusions and gives a discussion of future work.

Chapter 2

G2P conversion using decision trees

This chapter will introduce decision trees, on which all the following chapters will also be based. Decision trees were chosen for this project because they are, as a general classification technique, well-suited for adaptation to different uses, and also because they give good results and can easily be analysed.

Before discussing the training and optimisation of decision trees, methods will be discussed by which the graphemes and phonemes of a word may be aligned. After this, a detailed description will be given of the manner in which decision trees are trained. Lastly extensive tests will be carried out in order to determine the optimal parameters by which trees may be trained.

2.1 Symbol alignment

As discussed in Section 1.4, most G2P conversion techniques, including decision trees, require a one-to-one mapping between a word's phonemes and its graphemes (and their context). As very few dictionaries include manual alignments of this nature, it is necessary to develop an automatic algorithm to align the graphemes of the word's orthography and the phonemes of its pronunciation.

2.1.1 String lengths

The grapheme strings making up a word and the phoneme string corresponding to its pronunciation are, in general, neither the same length nor can they be aligned in a trivial manner. This is easily illustrated by a word such as *peace*, which is pronounced */p iy s/* - the first letter matches the first phoneme, but the following two letters together match a single phoneme, and the last letter doesn't match any phoneme at all. Fortunately, few graphemes align with more than one phoneme. Common exceptions in English include *x*, which commonly represents the two phonemes */k s/*, and *u*, which can represent */y uw/*, as in *use* [9]. Most systems described in the literature suggest that such situations can be solved by creating manually determined pseudophonemes, one representing each possible

pairing of two phonemes that must align to a single grapheme. In this project this was done by iteratively adding the pseudophonemes needed to align words' graphemes and phonemes until there existed a possible alignment for every word in the dictionary. By studying the pseudophonemes found in this way it was also possible to identify many irregular words, abbreviations and errors in the dictionary.

By creating pseudophonemes for each case where a single grapheme represents multiple phonemes, it is possible to continue on the assumption that graphemes align with at most a single symbol, either a phoneme or a pseudophoneme.

The alignment problem is thus reduced to optimally combining pairs of phonemes and inserting null phonemes, occasionally denoted by /ϵ/ [39], into the phoneme string so that it matches the length of the grapheme string [32]. Where groups of graphemes correspond to a single phoneme, the phoneme is assigned to the first of the graphemes and the rest are assigned null phonemes.

It is possible to build a system where, rather than a one-to-one mapping with the first grapheme in a group aligning to a phoneme and the rest to null phonemes, multiple graphemes correspond to a single phoneme or phonemes. Conceptually this shifts the alignment from having silent letters to having multiple letters that together form a single sound. There are situations where both approaches are meaningful, but the prevalence of silent letters in English (such as the *e* in *love*) serves as a reason to prefer the null phoneme interpretation - it allows all the situations where a given grapheme is silent to be handled similarly.

2.1.2 An iterative approach

The primary method of aligning graphemic and phonemic strings, and the only one that receives much attention in the literature, uses a form of the EM (Expectation-Maximisation) algorithm [32]. This is an iterative approach that alternately calculates the parameters of a system (from its expected values), and then maximises some property of the system using those parameters.

In the case of G2P conversion, the technique begins by estimating *P(G, P)*, the unigram probability that a given grapheme G matches a given phoneme P. Having done this, optimal alignments based on these probabilities are generated for all words in the corpus. The newly aligned words are then used to recalculate the unigram probabilities. These two steps are iteratively repeated until convergence.

Finding an optimal alignment given *P(G,P)* is accomplished using Dynamic Time Warping (DTW) [32], with the logarithm of the number of occurences of a grapheme-phoneme pair being used as the cost of matching graphemes and phonemes. The proof that this cost will give the optimal alignment is given in Appendix A.

The method proposed in [32] to initialise *P(G, P)* is to count grapheme/phoneme matches for all possible alignments of the training data. An alternative method, suggested in [14],

scores potential matches based on how close to the start of a word they occur. This somewhat simplistic approach is expanded in [35], where all possible matches are scored according to their distance from the diagonal, as illustrated in Figure 2.1. A third method, suggested in [2], uses only words where the graphemic and phonemic strings are already of equal length for initialising *P(G, P)*.

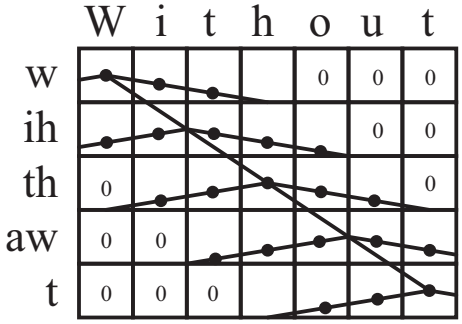


Figure 2.1: *Scoring matches based on their distance from a theoretical diagonal, representing a uniform distribution.*

2.1.3 Hand-seeding

A final modification which can be made to this algorithm is to create a set of possible grapheme-phoneme matches, and then constrain the algorithm to use only those mappings [32]. Creating such a list requires some human input, but it can be constructed with relatively little specific expert knowledge. This is done by starting with a simple table of a few known correspondences, and then iteratively finding any words in the dictionary for which no possible alignment exists and adding the necessary grapheme/phoneme matches to the list. Once such a list has been constructed the algorithm as described above can be used, with the restriction on which matches may be used. Pagel et al [32] report results that are slightly better when using such a hand-seeded approach.

This process was also very effective in identifying occasional errors in the dictionary used, as most errors result in pronunciations that cannot be aligned using the manually created set of allowed mappings.

2.2 Decision tree training

Training a decision tree is a recursive process, building the tree from the root node to the leaves. It starts by finding the best possible question for a node, given a set of training data corresponding to that node. Child nodes relating to the answers of the chosen question are then created. Finally the training data is split between those child nodes by applying the selected question. This whole process is then recursively repeated for each child node.

Two important concerns that arise during this process are how to determine the best question for a node and when to stop splitting nodes.

2.2.1 Question selection

In order to select a question, a set of potential questions must first be identified. This aspect is discussed later. To choose the optimal question, the information entropy of the node is calculated, and the total weighted entropy for all its children is calculated for each potential question. The information gain is then the difference between these two quantities. The question with the highest information gain is chosen [41].

If, for a general classification problem, there are n classes, each with probability p_i, the information entropy is given by Equation 2.1 [46].

$$H = -\sum_{i=1}^{n} p_i \log p_i \tag{2.1}$$

When applied to G2P decision trees, different phonemes supply the different classes, yielding the following formula:

$$i(t) = -\sum_{p} \frac{N_{t,p}}{N_t} \log\left(\frac{N_{t,p}}{N_t}\right)$$

In this equation, $i(t)$ is the information entropy of a node t, N_t is the number of training cases at the node, and $N_{t,p}$ is the number of occurraences of phoneme p at the node. Hence $\frac{N_{t,p}}{N_t}$ is an approximation of the probability of a phoneme at node t being of type p.

For determining the effect of a quesion on information entropy, entropy gain is used. For a node t with entropy $i(t)$, and with children t_L and t_R with respective entropies $i(t_L)$ and $i(t_R)$, and with proportions p_L and p_R of the parent node t's data divided among the respective children by a question, the entropy gain is given by [11] the following:

$$\Delta i = i(t) - p_L i(t_L) - p_R i(t_R)$$

Since we are searching for the question with the largest Δi, $i(t)$ can be omitted with no loss of generality as it remains constant for all questions at a given node. Furthermore p_k can again be approximated by N_{t_k}/N_t for a child node t_k. This gives the following expression for the entropy gain:

$$\begin{aligned}
\Delta i &= -\frac{N_{t_L}}{N_t}\left(-\sum_{p}\frac{N_{t_L,p}}{N_{t_L}}\log\left(\frac{N_{t_L,p}}{N_{t_L}}\right)\right) - \frac{N_{t_R}}{N_t}\left(-\sum_{p}\frac{N_{t_R,p}}{N_{t_R}}\log\left(\frac{N_{t_R,p}}{N_{t_R}}\right)\right) \\
&= \frac{1}{N_t}\left(\sum_{p} N_{t_L,p}\log\left(\frac{N_{t_L,p}}{N_{t_L}}\right) + \sum_{p} N_{t_R,p}\log\left(\frac{N_{t_R,p}}{N_{t_R}}\right)\right)
\end{aligned}$$

Finally, since $1/N_t$ is also constant for all questions, finding the question which maximises Equation 2.2 allows the optimal question to be found [22]:

$$\Delta i = \sum_p N_{t_L,p} \log\left(\frac{N_{t_L,p}}{N_{t_L}}\right) + \sum_p N_{t_R,p} \log\left(\frac{N_{t_R,p}}{N_{t_R}}\right) \tag{2.2}$$

2.2.2 Stop conditions

The most common conditions used to terminate the splitting of nodes are the placing of a minimum value on the information gain (this can be a minimum of zero), and the placing of a minimum on the number of training cases that are associated with any child node. Should a potential split result in too small an information gain, or leave insufficient training data in one of the child nodes, the node being examined is not split. In this case the node becomes a leaf, with the most common phoneme occurring in its training data taken as its class. In many decision tree applications this minimum quantity of training data is necessary to prevent overtraining and the associated deterioration of the tree's ability to generalise. Black et al [9], however, found that for G2P conversion this was not the case, and that the precision of their trees increased until there were leaves corresponding to single training examples.

2.2.3 Weighting

It is possible to further weight different training cases based on the frequency with which the words they are taken from occur in real-world texts. This favours more common words when building the tree. This approach is not often used, as it does not usually aid in the pronunciation of uncommon words. It can, however, be very useful when space limitations are a significant concern: by favouring common words, a smaller tree can perform as well as a more complete tree in most commonplace situations [41].

2.3 Questions and feature vectors

While specific questions for each node are chosen automatically based on their information gain, the overall set of possible questions must still be designed by hand. Within the context of G2P conversion, the most readily available features about which to ask questions are the neighbouring graphemes, and phonemes that have already been assigned to graphemes.

The simplest type of question is merely whether the grapheme or phoneme at a specific location corresponds to a specific symbol, or whether a word boundary occurs at a specific position (by treating the word boundary as a special type of symbol these become the same question). In addition to this, it is possible to ask questions about groups of symbols, e.g. *is the symbol at a certain position one of a given set of symbols*. Grouping symbols together in this way not only decreases the size of the tree, but can also improve its classification ability [22]. Intuitively one would be inclined to create such groups around the phonemic

nature of the symbols, such as grouping all vowels or all plosives together. Creating such rules by hand, however, is time-consuming and might not produce optimal groupings. It is thus preferable to generate such groupings, or *clusters*, automatically.

Kienappel and Kneser [22] describe a fairly detailed method of automatically creating such group questions. The first step is to cluster the graphemes or phonemes into a tree, a process which is explained in more detail in Section 2.3.1. Once the cluster trees have been generated, potential questions are generated for all nodes in the cluster tree, and the decision tree is grown as usual. In [22] it was also found that more distant graphemes have a smaller impact on the output, and as a result increasingly general questions were used with increasing context length to avoid noisy results.

The final consideration is how to resolve conflicts if multiple questions yield the same information gain. The method proposed is to use the following hierarchy [22]:

1. Prefer closer context
2. Prefer smaller grapheme/phoneme groups
3. Prefer non-phoneme questions

This ensures firstly that closer symbols are preferred, since closer graphemes and phonemes have a larger influence on the output phoneme. Secondly, smaller, or more specific clusters are preferred to avoid over-generalisation. Lastly, graphemic questions are chosen before phonemic ones because the graphemes are known with certainty, while phonemes are based on previous, potentially inaccurate classifications.

2.3.1 Clustering

There is no efficient way to optimally cluster graphemes or phonemes. The technique proposed by Kienappel and Kneser [22] is an approximation based on the following greedy algorithm:

```
place all symbols in the root node.
while there are leaf nodes with more than one symbol {
    for each such a node with more than one symbol {
        initialise single-symbol clusters for all symbols in the node
        while there are more than two such clusters {
            implement the merge causing the least gain in total entropy
        }
        move members between the two clusters to minimise entropy
        create a new child node for each of these two clusters,
                    and attach these two children to the current node
    }
}
```

This algorithm is applied independently for each focus grapheme and phoneme, so that separate cluster trees are generated for all possible context positions (the focus grapheme, neighbouring graphemes and previous phonemes). This ensures that, for each cluster tree, the training data consists of pairs of the context grapheme or phoneme, and the target phoneme.

The entropy is calculated based on the distribution of symbols within the node in question, according to Equation 2.1. The tree structure recursively splits the training data into nodes so that each node only contains the data corresponding to that node's graphemes or phonemes. The data within a node then provides an estimation of the relative probability distributions of the node's symbols. These probabilities can then be used to calculate the node's entropy.

While this algorithm is not guaranteed to find the optimal clusters, it has been found to produce sufficiently good clusters to reduce the size and improve the accuracy of the decision tree. Kienappel and Kneser do not, however, compare the algorithm's performance with that of hand-crafted clusters [22].

2.3.2 Pruning

It is common practice to grow decision trees to their maximum size, and then to reduce their size by pruning until some optimal point is reached. This optimum usually exists because very large trees will be biased in favour of the training data, and do not generalise well, while trees that are too small have a large variance and will not give accurate results. As a result of this tradeoff, an optimal tree size can be sought where the tree generalises well and still gives good predictions [11].

Pruning starts at the parents of the leaf nodes, and progresses back towards the root. For each node (and the subtree of which it is the root) four situations are considered using the portion of the pruning data associated with this node, as determined by the questions of nodes above it in the tree. These four situations are [39]:

1. The node, with its subtree, as it is
2. The node without either of its children
3. Replacing the node with its left child
4. Replacing the node with its right child

The best performing choice, based on the held-out pruning data, is then adopted. Performance is measured by determining the absolute number of classification errors made by the tree, with a threshold minimum improvement required to retain the child nodes.

Pruning is sometimes done using cost-complexity. In this case the cost-complexity is the quantity to be minimised. Cost-complexity is the sum of the number of classification errors and the product of a weighting factor α and the number of nodes in the resulting tree [11, 39].

2.4 Dataset

While there are a number of different English accents present in South Africa, first language speakers generally speak Standard South African English (SSAE), or what Bowerman refers to as 'White South African English' [10]. This accent is fairly similar to Received Pronunciation (RP) [10, 5], a prominent British English accent, but has a characteristic split of Wells' *KIT* vowel [49] into two clear allophonic variations, one close and one more central. Due to its prominence, especially among mother-tongue English speakers, it is this SAE accent that is used in this study.

SAEDICT, an SSAE pronunciation dictionary, was used to carry out experiments on the G2P conversion system. This dictionary is under development at Stellenbosch University, and is based on a frequency-based word list drawn from a general corpus of English text. All pronunciations in the dictionary were transcribed by the same linguistic specialist, ensuring their consistency. The transcriptions reflect the "correct", or commonly accepted pronunciation of an SSAE mother tongue speaker. For most entries there is only a single pronunciation, and variants have been included only in situations where they were deemed common and correct by the specialist. An example of such a word is *necessary*, which has a main pronunciation */n eh s ax s ax r ih/*, but the variant */n eh s ax s r iy/* is also included. Since the variants represent less standard pronunciations, they were discarded from the training data.

The version of the dictionary used contains 36 956 entries, and was taken from the live version on 1 April 2009. Of these a smaller subset of 23 031 was used for experimentation in this and the next chapter. This smaller set of words is common to three other dictionaries used for experimentation and introduced in Chapter 3. This subset was used in this and the next chapter to allow the experimental results to be directly comparable later.

Non-words, such as abbreviations and acronyms, and other words with very unusual pronunciations (such as *colonel*), were removed from the dictionary as they do not follow standard spelling rules and would therefore weaken the performance of the G2P converter. These words were identified during the alignment of the graphemes and phonemes, as no alignment could be found for them according to the allowed grapheme-phoneme correspondences.

SAEDICT is transcribed in ASTBET, a phoneme set based on IPA, developed with the annotation of South African languages in mind, and drawing on the experience of a number of such annotation projects [31]. It has a large number of phonemes, allowing for more

detail than many commonly used smaller phoneme sets. It includes sounds commonly found in SAE, as well as sounds found in or borrowed from other South African languages. In order to allow easy comparison with other English dictionaries, however, this was mapped to ARPABET [34], a phoneme set widely used in pronunciation dictionaries. A complete list of the ARPABET phoneme set used, as well as the mapping between ARPABET and ASTBET, can be found in Appendix B.

2.4.1 Training and testing data

Training and testing must necessarily both take place using a dictionary of words with known pronunciations. Since the same dictionary has to be used for both tasks, it is necessary to divide the dictionary into training and testing partitions. This division must be carefully considered, as both datasets must be properly representative of the dictionary (and thus hopefully the language) as a whole. Furthermore it is important that there is not too much overlap between the two, as might occur between words with the same root but different affixes. This should be avoided because it can cause the test data to be too similar to the training data, with the result that the test will not be properly indicative of the system's performance on completely unknown words. Error rates can drop by up to 60% as a result of training and test data containing words with overlapping roots [17]. If, however, the point of testing is merely to compare different systems, reasonably good comparative results should be obtained for any partition, as long as the same partitioning is used in all cases.

For the experiments reported in this and the next chapter, the lexicon was split into training and testing data in the ratio 90:10. After alphabetical sorting, sets of 90 and 10 consecutive words were placed into these respective sets (training and testing). The aim was to prevent most morphological variants from being split between the training and testing data, and thus provide a more accurate indication of the success of the system. In cases where decision tree pruning was used, the total training data was further divided into training and pruning data in the same way as the original training and testing split.

The frequency of the words was not directly used when training the system, as the primary goal of such a converter is to predict the pronunciations of uncommon words, and also because the most frequent words often have the least regular pronunciations.

2.5 Experimental results

The G2P conversion system developed has a number of parameters that can be adjusted so as to provide optimal performance. To achieve this, a number of tests were carried out to determine not only the best configuration for the system, but also to get an indication of its performance in general. The broad functioning of the system is illustrated in Figure 2.2.

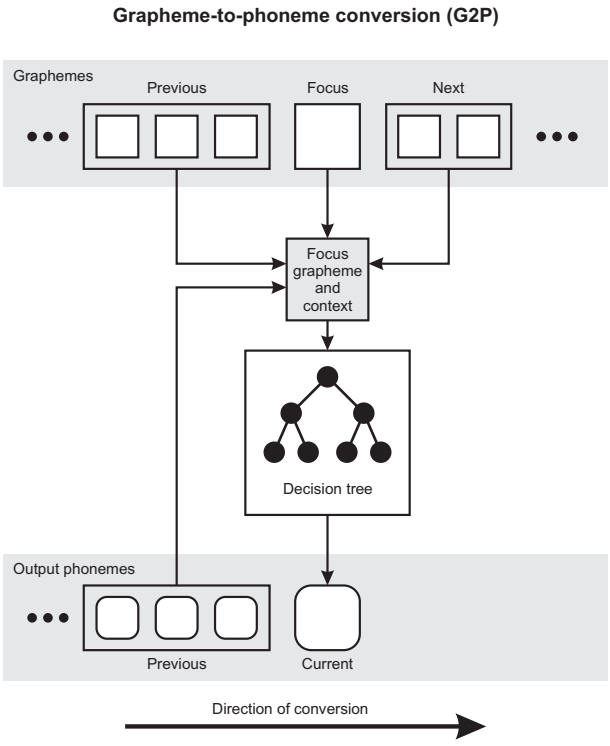


Figure 2.2: *Diagrammatic representation of G2P conversion using decision trees.*

All tests are conducted using the ARPABET phoneme set, except where clearly stated otherwise.

2.5.1 Accuracy measures

Results are given in terms of two different measures: phoneme accuracy and word accuracy. For both of these the generated phonemic transcription was first simplified by removing nulls and splitting double phonemes. The word accuracy shows how many entries have generated pronunciations exactly matching the correct pronunciation in the test dictionary. The phoneme accuracy is calculated as the number of correctly matching phonemes minus the number of phonemes inserted by the G2P conversion, as a fraction of the total number of phonemes in the correct transcription. This is given by the following equation:

$$Acc = \frac{N_{correct} - N_{insertions}}{N_{phonemes}} \tag{2.3}$$

In addition, 95% confidence intervals for the phoneme accuracy, and where applicable the probability of improvement (POI), were also calculated. The POI gives the probability, indicated as a percentage, that the result is an improvement over the preceding baseline value. Both the confidence intervals as well as the POI are calculated using the bootstrap method described in [7]. The former are determined by repeatedly sampling (with replacement) a random selection of the test data, calculating the mean for each sample, and then calculating the standard deviation of these means. The POI is calculated in a similar manner, but for each sample it is determined whether the modified system performs better than the baseline system.

All results are obtained using 10-fold cross validation, so as to minimise the effect of the specific data used for training and testing.

2.5.2 Alignment initialisation

An alignment of the phonemes and graphemes in the lexicon is required before the G2P conversion system can be trained and tested. As described in Section 2.1, there are a number of different ways of initialising these alignments. Furthermore, the alignments can be based purely on statistical methods, or can be seeded with a hand-constructed list of possible matches. All these options were explored by using the resulting aligned dictionary to train and test a decision tree. The results are shown in Table 2.1.

	Description	Phoneme	Word	Size
A	Equal length word/pronunciations only	90.86 ± 0.53%	60.20%	17602
B	Linear spread on diagonal (width 1)	90.74 ± 0.53%	59.81%	17672
C	Linear spread on diagonal (width 3)	90.79 ± 0.53%	59.94%	17652
D	All possible alignments (no doubles)	90.74 ± 0.53%	59.85%	17700
E	All possible alignments (with doubles)	90.63 ± 0.53%	59.34%	17537
F	All possible (hand-seeded)	90.80 ± 0.53%	60.04%	17490

Table 2.1: *Results using different alignment initialisations*

Phoneme refers to the phoneme accuracy, while *Word* refers to the word accuracy. *Size* is the number of nodes in the constructed decision tree.

Test A initialises the alignment matrix using only those words whose grapheme and phoneme strings have equal length, while tests B and C use weighting probabilities based on the distance of the alignment from the diagonal. Tests D, E and F all count each possible alignment of each word once. Test D doesn't allow double phonemes, while test E does (these were restricted to the minimum). Test F limits matches to a predefined set of grapheme-phoneme pairs, as given in Appendix C. This hand-seeded method was also evaluated in conjuction with all the other initialisation methods, but yielded almost identical results. As a result, the only hand-seeded test shown is test F, obtained using all possible alignments including double phonemes.

All the results were obtained by building decision trees that had context windows of three graphemes to the left and three to the right of the focus grapheme, and the three most recently determined phonemes. Pronunciations were constructed from right to left, and clusters used to construct group questions. These choices were found to be optimal in experiments that will be discussed in Sections 2.5.3, 2.5.4 and 2.5.6.

While using only words with equal length phonemic and graphemic strings during initialisation gave the best results, the differences between the various initialisation methods are not significant.

2.5.3 Phonemes and direction of processing

The experiments in this section determine whether adding previous phonemes to the tree improve performance, and if so, whether better results are obtained when words are processed from left to right or right to left. A context window extending three symbols to the left and three to the right was used, and clusters were used to determine questions as described in Section 2.3.1. Nodes in the decision tree were allowed to contain a minimum of one instance of the training data. As can be seen in Table 2.2, it is best to include phonemes and to move from right to left, as this results in a tree that is both smaller and more accurate.

2.5.4 Context window size

The objective of the experiments in this section is to determine the number of surrounding graphemes and phonemes (i.e. the window size) that leads to optimal G2P conversion performance. The remaining parameters were kept as before: phonemes and clusters were used, conversion took place from right to left, and training continued until nodes contained only a single instance of the training data. For right to left processing, only the right context can include phonemes as the left context has not yet been processed. Table 2.3 indicates that the system performs optimally for a symmetrical context window extending 3 symbols to the left and 3 to the right.

2.5.5 Minimum node size

The effect of varying the minimum number of training occurrences that must be associated with each node during decision tree training was also evaluated. The results, given in Table 2.4, confirmed the findings of other researchers, namely that the decision tree will not become overtrained and that splitting nodes up as far as possible will improve the system's performance [9]. However, the results also make it clear that it is possible to reduce the size of the trees with only a minimal negative impact on the system's accuracy.

	Phoneme	Word	Size	POI
No Phonemes	90.39 ± 0.52%	56.96%	18475	-
Phonemes, Left to Right	89.89 ± 0.55%	56.91%	17960	4.79%
Phonemes, Right to Left	90.87 ± 0.53%	60.26%	17604	92.68%

Table 2.2: *Results for including phonemes and direction of processing.*

Left	Right	Phoneme	Word	Size
1	1	84.30 ± 0.62%	37.48%	7145
1	2	88.04 ± 0.59%	50.91%	14508
2	1	82.31 ± 0.64%	32.46%	23670
2	2	90.02 ± 0.54%	56.89%	17315
2	3	90.73 ± 0.53%	59.44%	17229
3	2	89.00 ± 0.56%	53.25%	22103
3	3	90.87 ± 0.53%	60.26%	17604
3	4	90.71 ± 0.53%	59.69%	17440
4	3	90.47 ± 0.53%	58.77%	18767
4	4	90.59 ± 0.53%	59.18%	17887
5	5	90.39 ± 0.54%	58.73%	17891
6	6	90.25 ± 0.55%	58.29%	17736

Table 2.3: *Results for varying context window size.*

Minimum Node Size	Phoneme	Word	Size
1	90.87 ± 0.53%	60.26%	17604
2	90.28 ± 0.54%	58.02%	15212
3	90.30 ± 0.54%	58.23%	13257
4	90.25 ± 0.54%	57.86%	11755
5	90.15 ± 0.55%	57.55%	10628
10	89.70 ± 0.55%	56.31%	7194
20	89.05 ± 0.56%	53.59%	4657

Table 2.4: *Results for varying the minimum node size.*

2.5.6 Questions using clusters

Table 2.5 compares the performance of decision trees built with and without questions using clusters relating to groups of graphemes and phonemes, as discussed in Section 2.3.1. Using clusters improves the accuracy and also decreases the size of the tree. Tests were also run in which questions were limited to clusters of increasing size with increasing distance from the focus grapheme, but this caused a decrease in performance.

	Phoneme	Word	Size	POI
Without Clusters	90.62 ± 0.53%	59.32%	19032	-
With Clusters	90.87 ± 0.53%	60.26%	17604	78.79%

Table 2.5: *Results with and without clustering of graphemes and phonemes.*

2.5.7 Questions including the closest vowels

The decision tree questions can be extended to include the vowels nearest the focus grapheme. As proposed by Webster and Braunschweiler [48], the vowels serve as an approximation of the syllabic structure of the word, and are thus able to give additional information with which the decision tree can be grown. Unfortunately, as seen in Table 2.6, this did not improve the overall system performance.

	Phoneme	Word	Size	POI
Without Vowels	90.87 ± 0.53%	60.26%	17604	-
With Vowels	90.56 ± 0.54%	59.41%	18830	26.55%

Table 2.6: *Results with and without including vowels in context.*

2.5.8 Questions regarding generated null phonemes

A novel feature was tested as a source of potential questions. This involved adding to the context both the number of preceding null phonemes and the previous non-null phoneme generated. Table 2.7 shows that this increased the system's accuracy, but only very slightly.

	Phoneme	Word	Size	POI
Without nulls	90.84 ± 0.53%	60.10%	17656	-
With nulls	90.87 ± 0.53%	60.26%	17604	63.21%

Table 2.7: *Results with and without questions about generated null phonemes.*

2.5.9 Other features

Lastly two other features were tested as potential questions. The first included the first and last graphemes in the word, the second the distance from the focus grapheme to the beginning and end of the word. Table 2.8 shows that these question types did not increase the system's accuracy.

	Phoneme	Word	Size	POI
Baseline	90.87 ± 0.53%	60.26%	17604	-
With first & last grapheme	90.13 ± 0.54%	57.46%	19195	0.07%
With start/end distance	90.34 ± 0.54%	58.41%	19384	0.86%

Table 2.8: *Results with and without first and last graphemes and distance to the beginning and end of the word.*

2.5.10 Pruning

It is common practice to explicitly prune decision trees using a held-out set of pruning data. Tests were run using cost-complexity pruning, as described in Section 2.3.2. Optimal results were obtained with a weighting factor α of 0 and a threshold of 3. This means that the tree size did not influence the pruning process, and that a decrease of at least three in the number of classification errors was required before a branch was pruned.

A held-out pruning set was obtained by splitting the total training data. The results, shown in Table 2.9, show that the best performance was obtained when the entire dictionary was used for training, i.e. when pruning was not carried out. It should be noted, however, that pruning reduces the tree size with minimal change in accuracy and could thus be particularly useful in situations where storage is limited.

	Phoneme	Word	Size
Without Pruning	90.87 ± 0.53%	60.26%	17604
5% Pruning, 95% Training	90.11 ± 0.55%	57.80%	15849
10% Pruning, 90% Training	89.92 ± 0.55%	57.07%	14426
15% Pruning, 85% Training	89.95 ± 0.55%	56.93%	13294
20% Pruning, 80% Training	89.76 ± 0.55%	56.14%	12282

Table 2.9: *Results with and without pruning the decision tree.*

2.5.11 Phoneme set

ASTBET, the phoneme set in which SAEDICT is transcribed, is quite large, with 59 phonemes used for SAE. ARPABET, in contrast, has only 47 phonemes. Comparative tests were run on SAEDICT using both phoneme sets. As shown in Table 2.10, the smaller phoneme set led to slightly higher accuracies, but much larger decision trees.

	Phoneme	Word	size
ASTBET	88.11 ± 1.17%	50.52%	5809
ARPABET	90.87 ± 0.53%	60.26%	17604

Table 2.10: *Results using two different phoneme sets.*

In order to allow for direct inter-accent comparison, as described in Section 2.4, the experiments in Chapter 3 will be carried out using ARPABET.

2.5.12 Phoneme analysis

A finer analysis of specific prediction errors showed no unexpected behaviour. A condensed confusion matrix of incorrectly predicted phonemes is given in Appendix D. Phonemes that are often incorrectly predicted are usually vowels, often those for which there is very little training data. By far the most problematic vowel is the schwa, /*ax*/, which is frequently predicted incorrectly. It is also subject to a number of deletions and insertions. Consonants being readily confused are /*dh*/ and /*th*/, /*g*/ and /*jh*/, and /*s*/ and /*z*/.

Table 2.11 gives the error rates for vowels and consonants, listed according to the frequency with which they occur in the dictionary. It is clear that consonants can be predicted more accurately than vowels. Only 6 consonants have accuracies below 90%, whereas all but one vowel are predicted with less than 90% accuracy. This reflects to a large extent the ambiguity of English orthography. Of the 26 letters in the alphabet, only 5 or 6 are used to represent vowels, while the remainder represent consonants. In spite of this, there are almost as many vowel phonemes as there are non-vowel phonemes – in its original ASTBET phoneme set, SAEDICT uses 29 vowels and diphthongs, and 30 consonants, and once

Consonants			Vowels		
Phoneme	Freq (%)	Accuracy (%)	Phoneme	Freq (%)	Accuracy (%)
t	6.97	98.2	ax	10.86	80.5
s	6.94	93.7	ih	8.77	85.6
n	6.84	98.9	eh	2.94	80.7
l	5.46	99.6	aa	2.47	78.9
d	4.69	98.8	ae	2.14	81.2
r	4.68	99.0	ey	1.88	84.2
k	4.34	98.5	ah	1.42	84.5
m	3.01	100.0	iy	1.36	71.9
p	3.00	98.8	ay	1.33	80.4
z	2.71	86.5	ow	1.15	77.1
b	1.97	99.1	uw	0.93	81.6
ng	1.80	97.9	ao	0.90	87.5
f	1.70	98.9	er	0.66	82.6
sh	1.31	95.8	uh	0.60	61.2
v	1.31	99.4	aw	0.37	82.5
g	1.12	92.8	oy	0.15	95.6
w	0.88	89.8			
y	0.88	72.9			
jh	0.79	94.4			
hh	0.70	95.3			
ch	0.44	83.0			
th	0.34	93.0			
dh	0.11	73.3			
zh	0.09	75.4			

Table 2.11: *Phoneme frequency and accuracy.*

mapped to the simpler ARPABET, of which not all the phonemes occur in SAEDICT, it uses only 16 vowels and diphthongs, and 24 consonants. As a result, consonants enjoy an almost one-to-one mapping with graphemes, while this is strongly not the case for vowels.

The table also shows a slight downward trend in the accuracy with decreasing frequency. This may be ascribed to the fewer training cases associated with less common phonemes.

2.5.13 Grapheme analysis

An approximate analysis of the accuracy with which the phonemes associated with each grapheme is predicted can be carried out using the alignment between graphemes and phonemes which is implicit in the training process.

Consonants		Vowels	
Grapheme	Accuracy (%)	Grapheme	Accuracy (%)
x	91.0	o	81.3
s	91.8	u	82.5
j	92.3	a	82.8
z	92.9	e	84.6
w	93.6	i	86.0
h	96.2	y	90.6
r	96.4		
g	96.5		
c	96.8		
t	97.1		

Table 2.12: *Grapheme performance in order of increasing accuracy.*

A summary of the results is given in Table 2.12. All graphemes not shown have accuracies greater than 98%. All consonants perform considerably better than vowels, with the semi-vowel 'y' falling in between these two categories. This has also been found in the previous section, where vowel phonemes are associated with higher error rates.

2.5.14 Word analysis

In order to determine whether proper nouns present a particular challenge to G2P conversion, a separate test was carried out in which the system was trained using the entire dictionary, and then tested first on only proper nouns, and then on the remaining words in the test set. It was found that the performance was slightly worse for proper nouns, with a phoneme accuracy approximately 4% lower and a word accuracy 5% lower than the remaining words. Of the 23 031 entries used for experiments in this chapter, 6 081 are proper nouns.

The dictionary contains too few foreign words to provide a meaningful analysis of the system's ability to predict their pronunciation.

The frequency with which words occur in the corpus from which the dictionary's words were drawn was also considered. Words in the test data were divided into blocks by frequency, and the word accuracy of each block calculated.

Occurrences	Words	Accuracy
1	5223	61.3%
2	3235	63.6%
3	2294	62.9%
4-5	2719	64.7%
6-8	2277	63.5%
9-13	2004	64.5%
14-25	2014	66.5%
26-76	2009	63.8%
77-66537	1096	65.2%
All	22871	63.6%

Table 2.13: *Word accuracy for different word frequencies.*

The results are given in Table 2.13. The first column, *Occurrences*, indicates the number of times words occurred in the text corpus from which the word list was drawn. *Words* shows the number of words of the given block, and *Accuracy* gives the word accuracy for those words. Results are listed from least to most frequent. The total over all the testing data is given at the end of the table.

The results reveal that the least frequent words perform slightly worse than other, more frequent words. Besides the least frequent block, however, there is no clear relationship between accuracy and frequency.

2.6 Chapter summary

This chapter began with a detailed description of the decision tree algorithm used for G2P conversion, along with the supporting algorithms that allow alignment, clustering and pruning. Various possible questions for the decision tree were also discussed.

The remainder of the chapter contained the results of extensive tests, with the aim of finding the optimal parameters for G2P conversion using decision trees, and also of investigating the performance of the system using these parameters. It was found that the best performance was obtained using a context window of three graphemes to either side of the focus grapheme, as well as the three previously generated phonemes and null questions relating to these phonemes. Clusters were used to obtain group questions. Pruning the tree did not yield improved performance.

The next chapter will extend the algorithms developed here so that they can be applied to the conversion of pronunciations between accents.

Chapter 3

Accent conversion

As mentioned in Section 1.3, the development of pronunciation dictionaries involves a great deal of time and effort by linguistic experts. This may be prohibitively expensive for under-resourced languages and accents such as the various SAE accents. Nevertheless, phonemic transcriptions in the speaker's accent can greatly improve both ASR accuracy [20] and TTS quality [16]. G2P conversion is often used to obtain the pronunciations of words not yet in the dictionary, but this technique has limited accuracy and has been shown to require a large training set.

This chapter considers the pronunciations of corresponding words in the General American (GenAm), Received Pronunciation (RP) and SSAE accents of English. The first two are commonly-used and widely studied reference accents for American and British English respectively [49].

From our perspective the objective of accent conversion is to determine whether British or American pronunciations can be used to complement our still fairly small SSAE pronunciation dictionary, and how best this can be done. However, the techniques we consider are generally applicable to any accent pair.

3.1 Dictionaries

Four dictionaries were used to represent the three English accents. CMUDICT [13] and PRONLEX [24] were used for GenAm pronunciations and BEEP [4] for RP. SAEDICT was used as a source of SSAE pronunciations.

CMUDICT, developed at Carnegie Mellon University (CMU), has words drawn from a variety of sources, including an initial subset developed by hand at CMU and extensively proofed and used. Its other sources include the Shoup dictionary and part of the Dragon dictionary.

PRONLEX, the LDC's COMLEX English Pronouncing Lexicon, was hand transcribed under direction of Cynthia McLemore at the Linguistic Data Consortium, and although multiple people were involved in its development, efforts were taken to ensure consistency,

and transcriptions chosen to reflect a "standard" American pronunciation.

The British English Example Pronunciation (BEEP) Dictionary is drawn primarily from the MRC Psycholinguistic Database and the Computer Usable Oxford Advanced Learners Dictionary. In addition to these two dictionaries, it has pronunciations provided by sources in Durham and Oxford Universities.

Table 3.1 describes each dictionary in terms of the number of words it contains, the total number of pronunciations and the average number or pronunciations per word (P/W). The number of pronunciations and P/W for the 23 031 common words described in Section 3.1.2 are also shown.

Dictionary	Entire dictionary			Common set	
	Words	Prons	P/W	Prons	P/W
SAEDICT	30 390	36 954	1.22	28 087	1.22
BEEP	237 592	256 998	1.08	25 576	1.11
CMUDICT	123 292	133 694	1.08	26 771	1.16
PRONLEX	90 919	99 978	1.11	25 471	1.11

Table 3.1: *Number of entries in the four dictionaries.*

3.1.1 Phoneme set

ARPABET [34] was chosen as the common phoneme set in which to analyse the four dictionaries. Both BEEP and CMUDICT already use ARPABET, while PRONLEX uses an ARPABET short-hand that can easily be converted to standard ARPABET. As described in Section 2.4, SAEDICT is transcribed in ASTBET. This was converted to ARPABET by means of a mapping based on the closest IPA symbol. A small number of phonemes were found to be present in only a single dictionary, and these were mapped to alternative phonemes as indicated in Table 3.2. This normalisation ensures that the phoneme sets used by the four dictionaries match exactly.

Dictionary	Mapping
BEEP	*/oh/* ⇒ */aa/* */ia/* ⇒ */ih ax/* */ea/* ⇒ */eh ax/* */ua/* ⇒ */uh ax/*
PRONLEX	*/wh/* ⇒ */w/* */en/* ⇒ */ax n/*

Table 3.2: *Phoneme mappings used to achieve a common phoneme set.*

3.1.2 Word list

In order to compare pronunciations, the set of words common to all four dictionaries was determined. Before extracting this set, all words were converted to standard UK spelling [47]. The final set of common words contained 23 031 entries.

There are a number of words in each dictionary with multiple pronunciations, either homographs (words with the same spelling but different pronunciations), or words with alternate pronunciations. The number of pronunciations vary, but all four dictionaries contain an average of between 1.11 and 1.22 pronunciations per word, as reflected in Table 3.1. Of the 23 031 words, 9 344 have multiple pronunciations in at least one of the dictionaries.

3.2 Pronunciation alignments

The pronunciations of corresponding words in the three accents were aligned using the dynamic programming (DP) algorithm described in Section 2.1. This was done both so that accents could be compared directly, and because a one-to-one correspondence between phonemes in different accents is needed for inter-accent conversion using decision trees (described in Sections 3.6 and 3.7).

An example of such an alignment is shown in Table 3.3. Nulls have been introduced where a phoneme has no counterpart in the other string (i.e. insertions and deletions). In this case the */ax/* in the SAE pronuncation is deleted in RP.

SAEDICT	*r*	*ih*	*ae*	*k*	*sh*	*ax*	*n*	*s*
BEEP	*r*	*ih*	*ae*	*k*	*sh*	-	*n*	*z*

Table 3.3: *Two aligned pronunciations of* reactions.

Words with multiple pronunciations pose a problem both when comparing two dictionaries, and when subsequently using the aligned pronunciations to train decision trees for accent conversion. This problem was solved by using only the single pair of pronunciations per word (one from each dictionary) that gave the best alignment.

Preliminary experiments were also conducted by considering all the possible pairings of pronunciations between the two dictionaries. The results indicated that this did not influence the relative accuracies obtained for the various accents, however, and hence the simpler approach of using the best-aligned pair was chosen for all further experiments.

3.3 Direct phonetic comparison

After alignment, the four dictionaries' pronunciations were first compared directly. The results are summarised in Table 3.4. The target dictionary, which is listed second in Table

3.4, is used as a reference against which the source dictionary, listed first, is tested. Because insertions for a dictionary pair in a given direction correspond to deletions when the pair is reversed, the results are asymmetrical.

The phoneme accuracy was again calculated according to Equation 2.3. This accuracy is given for all phonemes, for vowels and for consonants. The last column indicates the percentage of words that have identical pronunciations.

	Phoneme	Vowel	Cons	Word
BEEP ⇒ SAEDICT	92.1	86.2	95.6	59.6
CMUDICT ⇒ SAEDICT	81.6	60.2	94.8	29.4
PRONLEX ⇒ SAEDICT	87.3	73.3	95.8	43.0
SAEDICT ⇒ BEEP	92.2	82.7	98.2	59.6
CMUDICT ⇒ BEEP	84.8	67.1	95.5	37.8
PRONLEX ⇒ BEEP	89.5	77.8	96.6	50.6
SAEDICT ⇒ CMUDICT	81.6	59.4	95.5	29.4
BEEP ⇒ CMUDICT	84.5	69.2	93.7	37.8
PRONLEX ⇒ CMUDICT	89.0	73.2	98.6	50.7
SAEDICT ⇒ PRONLEX	87.2	72.3	96.5	43.0
BEEP ⇒ PRONLEX	89.3	80.2	94.7	50.6
CMUDICT ⇒ PRONLEX	89.0	73.2	98.6	50.7

Table 3.4: *Accuracies (%) of direct pronunciation alignments between dictionaries.*

It is clear from Table 3.4 that SAEDICT and BEEP match most closely, while the poorest correspondence is found between CMUDICT and SAEDICT. On average, the American accents are closer to RP than to SSAE. Furthermore, vowels differ more strongly between accents than consonants. While at least 93.7% of consonants matched between any pair of dictionaries, this figure varied between 59.4% and 86.2% for vowels.

3.4 Phoneme shifts

RP, GenAm and to a lesser extent SSAE are well-documented in phonetic and linguistic literature. Wells [49] especially gives a detailed analysis of the different accents. This was compared to the phoneme confusion statistics obtained from the alignments determined in Section 3.3.

3.4.1 Consonants

There is essentially no difference in the consonant systems of the three accents [23], as confirmed by Table 3.4. There are, however, a few differences in the frequency and environments in which certain consonants are used.

- GenAm is a rhotic accent, whereas RP and SAE are non-rhotic [49]. Rhotic accents articulate the *'r'* in non-prevocalic contexts such as *farm* and *far*, which is silent in non-rhotic accents. In GenAm these words are pronounced */f aa r m/* and */f aa r/*, while in SSAE and RP they are */f aa m/* and */f aa/*. In the alignments, some 17% of */r/* phonemes in PRONLEX and CMUDICT are deleted in BEEP and SAEDICT.

- The use of the semi-vowel */y/* after a consonant varies, for example in *tune*, *duke*, *new* and *temperature*. SSAE uses this phoneme in almost all cases. RP uses */y/* in most cases, except after */t/* where it sometimes merges to form a */ch/*. GenAm drops the */y/* in all contexts except after labials and velars, such as *cute* and *beauty*. This phenomenon, described by Wells [49] as Later Yod Dropping, manifests itself in the confusion statistics: approximately 25% of */y/*'s in SAEDICT and BEEP are deleted in CMUDICT and PRONLEX.

- The sibilants */s/* and */z/* have a slightly different distribution in SSAE when compared with the other two accents: words like *holds* and *levels* contain */s/* rather than */z/*. This is especially prevalent in word-final positions. This change, not mentioned by Wells, occurs for approximately 16% of */z/* sounds in RP and GenAm. A related shift occurs for */dh/*, where some 15% are substituted with */th/* in SSAE. Examples include *earthenware*, *wreaths* and *baths*.

- RP uses a syllabic consonant for */l/* and */n/* in words like *bubble* or *sudden*, while GenAm and SSAE use a schwa and a consonant, that is */ax l/* and */ax n/* [49]. Of the schwas in SAEDICT, 15% are deleted in BEEP (with comparable results between PRONLEX and BEEP). Of these deletions, almost all occur before */n/* or */l/* (51% and 45% respectively in SAEDICT).

3.4.2 Vowels

Vowels exhibit a much larger and less systematic pronunciation variation than consonants. There are, however, certain well-known differences.

- The words *cot* and *caught* are pronounced in the same way in GenAm, but not in SSAE or RP [49, 23]. This reflects what Wells terms the THOUGHT/LOT merge, where the vowels present in these words have merged in American speech. Our analysis finds that about 12% of */aa/* phonemes in SAEDICT and BEEP become */ao/* in CMUDICT and PRONLEX. A related shift is the approximately 11% of */aa/*'s in BEEP and SAEDICT that become */ae/* in GenAm, such as the vowel in *can't*.

- SSAE uses */ih/*, primarily at the end of words, for example *happy* and *cheeky*, but also in words like *barrier*, while other accents use the higher vowel */iy/* [49]. More than 55% of the */iy/* phonemes in the other accents map to */ih/* in SAEDICT. This corresponds to the *KIT* split mentioned in Section 2.4.

- Other variations within vowel correspondences are mostly related to the schwa, */ax/*. These shifts show no obvious structure, but seem to reflect the accents' different stress patterns, as unstressed vowels are frequently weakened to */ax/*.

3.5 G2P conversion for the four dictionaries

G2P conversion was applied to each dictionary individually, to gauge how accurately pronunciations could be produced based on training data drawn from the same accent. This was done using the same parameters and test conditions described in Chapter 2. The context included 3 graphemes to either side, the three previous phonemes, and null questions. Clusters were used to generate questions about groups of symbols. Of the 23 031 words available in each dictionary, 90% were used for training and 10% for testing. 10-Fold cross-validation was then applied to minimise the effect of the training/test data split.

Dictionary	Phoneme	Word	Size
SAEDICT	90.87 ± 0.52%	60.26%	17604
BEEP	91.61 ± 0.53%	64.55%	14331
CMUDICT	91.33 ± 0.54%	63.72%	15300
PRONLEX	92.24 ± 0.51%	66.16%	14016

Table 3.5: *G2P accuracies for the four dictionaries, with 95% confidence intervals and decision tree sizes.*

As shown in Table 3.5, there is little variation among the three accents. However, it is clear that SAEDICT has the lowest G2P accuracy, while PRONLEX has the highest. This suggests that SAEDICT has the least regular relationship between graphemes and phonemes. All the results are comparable to those found by other researchers, where accuracies of approximately 91% are common [9, 39].

3.6 P2P conversion

Grapheme-to-phoneme conversion can be used to estimate the pronunciations of new words, given a set of training pronunciations (i.e. a pronunciation dictionary) in the same accent or language. It is sometimes the case, however, that pronunciations for the desired words are already available, but in a different accent of the same language. For example, the pronunciations of many words currently not in SAEDICT are available in CMUDICT, PRONLEX or BEEP. We now address the question of whether such existing pronunciations in a different (source) accent can be used to estimate the pronunciation in the desired (target) accent. Since this process involves the conversion of one phoneme string (the source pronunciation)

to another (the target pronunciation), it will be referred to as phoneme-to-phoneme (P2P) conversion.

The same decision tree algorithm used for G2P conversion will be used for P2P conversion, with the phonemes of the source accent taking the role of the "graphemes" and those of the target accent as the "phonemes". This process is shown diagrammatically in Figure 3.1.

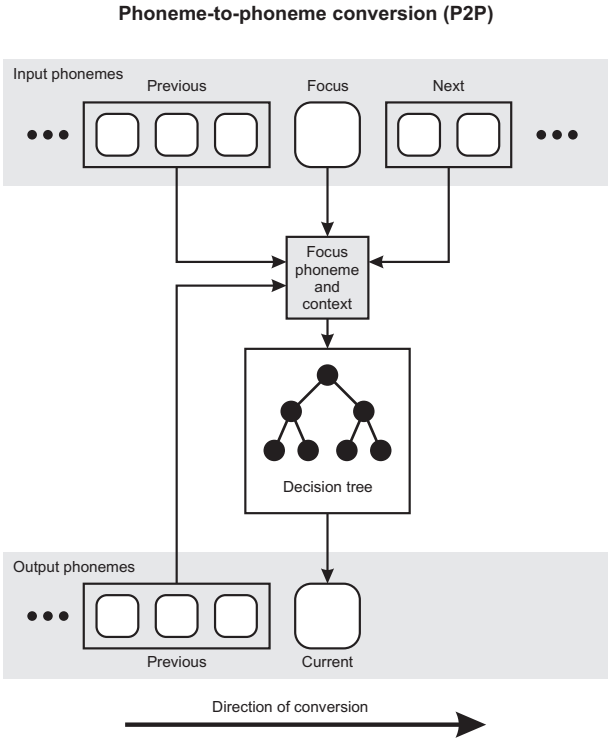


Figure 3.1: *Diagrammatic representation of P2P conversion using decision trees.*

A notable difference between G2P and P2P conversion is a higher incidence of insertions when aligning two pronunciations. While there are relatively few cases of a single grapheme corresponding to multiple phonemes, there are considerably more cases of a single source phoneme corresponding to multiple target phonemes. This was dealt with by allowing the decision tree to use a greater variety of pseudophonemes (double phonemes treated as a single symbol).

3.6.1 Decision tree parameters

As for G2P in Chapter 2, tests were carried out to determine the optimal decision tree parameters for P2P conversion. These were carried out for the specific case of conversion from BEEP to SAEDICT. However, the same conclusions were reached using other dictionary combinations. As with G2P, processing was done from right to left. A summary of the results of these tests is given in Tables 3.6 and 3.7. It is clear from these results that optimal performance is obtained using one phoneme to the left and two to the right as context.

Furthermore, pruning the tree using part of the data yielded improved results, unlike for G2P. The optimal split was to use equal portions of the data for training and pruning the tree respectively.

Left	Right	Phoneme	Word	Size
1	1	95.17 ± 0.35%	74.59%	2647
1	2	95.42 ± 0.35%	76.06%	3352
2	1	94.86 ± 0.37%	73.28%	3806
2	2	95.36 ± 0.35%	75.84%	3975
2	3	85.35 ± 0.35%	75.73%	3868
3	2	85.25 ± 0.36%	75.29%	3967
3	3	95.30 ± 0.35%	75.34%	3764

Table 3.6: *P2P results for varying context window size, using 50% pruning and 50% training data.*

	Phoneme	Word	Size	POI
Without Pruning	95.21 ± 0.36%	74.92%	8180	Baseline
20% Pruning, 80% Training	95.30 ± 0.35%	75.43%	5068	58.08%
45% Pruning, 55% Training	95.35 ± 0.35%	75.71%	3801	76.51%
50% Pruning, 50% Training	95.42 ± 0.35%	76.06%	3352	87.78%
55% Pruning, 45% Training	95.39 ± 0.35%	75.85%	3132	82.15%

Table 3.7: *P2P results for different levels of decision tree pruning, using a context window of 1 phoneme to the left and 2 to the right.*

3.6.2 Comparison with hand-crafted rules

A comparison was carried out of the P2P conversion technique developed in Section 3.6, and hand-crafted rules developed by the Meraka Institute to convert pronunciations from OALD to SAEDICT [25]. These rules were designed on the basis of the available phonetic literature for South African English [49, 10, 5]. OALD [30] is one of the main sources of BEEP, so the P2P conversion system could easily be modified to use it.

	Phoneme	Word
Automatically trained decision trees	96.03 ± 0.33%	79.20%
Hand-crafted rules	93.89 ± 0.38%	68.33%

Table 3.8: *Comparison of P2P conversion using automatically trained decision trees and using hand-crafted rules.*

It is clear from the results in Table 3.8 that the automatically trained decision trees give much more accurate P2P conversion than the hand-crafted rules. A further analysis of these differences is the subject of ongoing research.

3.6.3 P2P results

Tests were carried out to determine the P2P conversion accuracy between all pairs of dictionaries, and the results are presented in Table 3.9. The phoneme and word accuracies are again calculated as described in Section 2.5.1. The same 90:10 split between training and testing data was used to allow direct comparison with the performance of G2P conversion.

	Phoneme	Word	Size
BEEP ⇒ SAEDICT	95.42 ± 0.35%	76.06%	3352
CMUDICT ⇒ SAEDICT	93.83 ± 0.40%	68.64%	4779
PRONLEX ⇒ SAEDICT	94.70 ± 0.37%	72.40%	3979
SAEDICT ⇒ BEEP	95.12 ± 0.37%	75.46%	3460
CMUDICT ⇒ BEEP	94.35 ± 0.40%	72.08%	4271
PRONLEX ⇒ BEEP	95.29 ± 0.37%	76.45%	3513
SAEDICT ⇒ CMUDICT	93.27 ± 0.44%	67.57%	4560
BEEP ⇒ CMUDICT	94.03 ± 0.43%	71.43%	4363
PRONLEX ⇒ CMUDICT	96.66 ± 0.31%	82.08%	2586
SAEDICT ⇒ PRONLEX	93.95 ± 0.42%	69.97%	4200
BEEP ⇒ PRONLEX	94.65 ± 0.40%	73.60%	3928
CMUDICT ⇒ PRONLEX	95.95 ± 0.33%	78.10%	3094

Table 3.9: *P2P conversion accuracies between accent pairs, with 95% confidence intervals and decision tree sizes.*

It is clear that the pronunciations that can most accurately be derived from each other are CMUDICT and PRONLEX. This suggests that the apparent differences when directly comparing their pronunciations (Table 3.4) are at least in part due to systematic differences in the transcription conventions followed, rather than fundamental differences in the pronunciations.

Overall, the accuracy of P2P conversion is also considerably higher than that of G2P conversion. A detailed discussion of these results is, however, deferred to Section 3.8.

3.7 GP2P conversion

Some words with different spellings have identical pronunciations in one accent but not in another. An example of such a pair of words is *spokesman* and *spokesmen*. In BEEP both are pronounced */s p ow k s m ax n/*, while in SAEDICT the latter is pronounced */s p ow k s m eh n/*. Table 3.10 indicates how many such pronunciations there were for each dictionary pair. These words have identical pronunciations in the source but not in the target dictionary. It has also been observed that there are generally fewer phonemes than graphemes in a dictionary entry. These considerations suggest that there may be information present in the orthography that is not represented phonemically. This hypothesis is tested in this section by using both the graphemes and the phonemes of the source accent to derive the pronunciation in the target accent.

	to SAEDICT	to BEEP	to CMU.	to PRON.
from SAEDICT	-	157	197	183
from BEEP	145	-	160	147
from CMUDICT	162	171	-	63
from PRONLEX	166	168	68	-

Table 3.10: *Number of source pronunciations with different graphemes and target pronunciations.*

This modification will be referred to as grapheme-and-phoneme-to-phoneme conversion (GP2P). In order to employ the source graphemes as additional context for the decision tree, the source phonemes and source graphemes were first aligned as described in Section 2.1. This alignment was then reversed, so as to associate a sequence of zero or more graphemes with every source phoneme (rather than associating zero or more phonemes with each grapheme). Figure 3.2 illustrates the GP2P conversion process.

The decision tree question set was extended to include questions of the type "does the grapheme sequence aligned with the phoneme contain the letter X". These questions could be asked for the focus phoneme or any phoneme in the input context. Preliminary experiments indicated that basing questions on clusters of graphemes obtained in a similar way to the input phoneme clusters did not lead to any performance improvements. Including the length of the grapheme sequence aligned with a source phoneme, or matching the sequence exactly to one of a list of known sequences, also did not lead to any improvements. These results are shown in Table 3.11. The first test includes only the first type of question mentioned, with other tests adding question types.

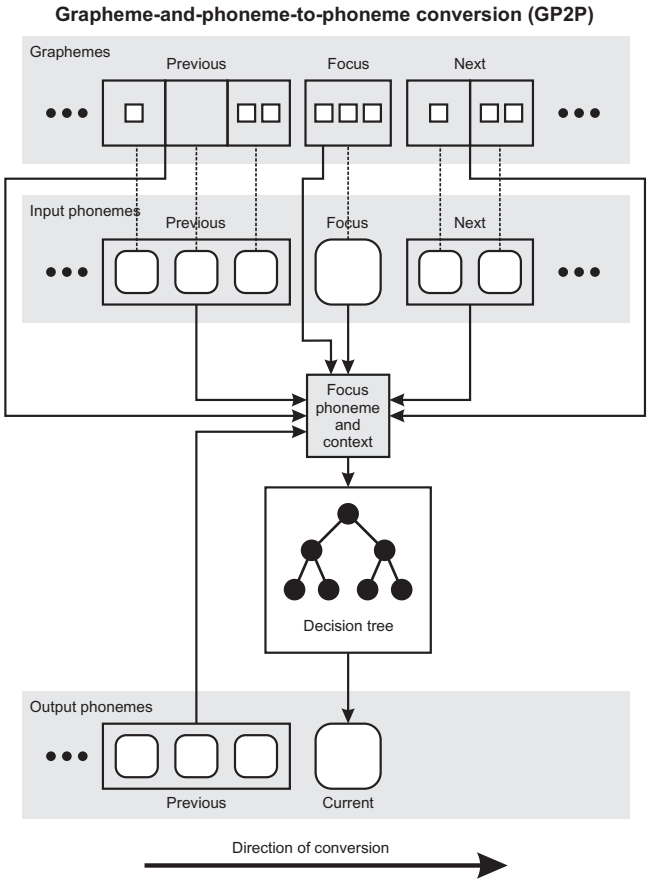


Figure 3.2: *Diagrammatic representation of GP2P conversion using decision trees.*

3.7.1 Decision tree parameters

Tests were run to determine optimal parameters for GP2P trees. Besides the additions to the question set, the only variation to those used for P2P conversion is the best training-pruning split. As shown in Table 3.12, it is marginally better to train with 45% and prune with 55% of the total training dictionary, whereas for P2P the best results came from training and pruning sets of equal size. The improvement is not significant, however.

	Phoneme	Word	Size
Basic grapheme sequence questions only	96.66 ± 0.30%	82.05%	2535
With clusters of graphemes	96.57 ± 0.30%	81.57%	2520
With exactly matching grapheme sequences	96.61 ± 0.31%	81.89%	2395
With grapheme sequence length	96.57 ± 0.30%	81.61%	2469

Table 3.11: *GP2P results for different types of questions regarding the grapheme sequences.*

	Phoneme	Word	Size
50% Pruning, 50% Training	96.60 ± 0.30%	81.61%	2710
55% Pruning, 45% Training	96.66 ± 0.30%	82.05%	2535

Table 3.12: *GP2P results for different levels of pruning.*

3.7.2 GP2P results

Tests were carried out to determine the GP2P conversion accuracy between all pairs of dictionaries, and the results are given in Table 3.13.

	Phoneme	Word	Size
BEEP ⇒ SAEDICT	96.66 ± 0.30%	82.05%	2535
CMUDICT ⇒ SAEDICT	95.70 ± 0.34%	77.24%	3274
PRONLEX ⇒ SAEDICT	96.05 ± 0.32%	78.88%	3011
SAEDICT ⇒ BEEP	97.40 ± 0.28%	86.59%	2203
CMUDICT ⇒ BEEP	96.63 ± 0.32%	82.91%	2802
PRONLEX ⇒ BEEP	97.10 ± 0.30%	85.10%	2439
SAEDICT ⇒ CMUDICT	96.47 ± 0.32%	81.74%	3018
BEEP ⇒ CMUDICT	96.80 ± 0.32%	83.80%	2660
PRONLEX ⇒ CMUDICT	97.77 ± 0.26%	87.94%	1915
SAEDICT ⇒ PRONLEX	97.36 ± 0.28%	85.64%	2400
BEEP ⇒ PRONLEX	97.42 ± 0.28%	86.31%	2289
CMUDICT ⇒ PRONLEX	97.83 ± 0.24%	87.79%	1949

Table 3.13: *GP2P conversion accuracies between accent pairs, with 95% confidence intervals and decision tree sizes.*

A discussion of the relationship between the GP2P, P2P and G2P results is deferred to the next section.

3.8 Discussion

When comparing Tables 3.5 and 3.9, it is clear that the pronunciation of a word in a new accent can be substantially more accurately determined from its pronunciation in another accent by means of P2P conversion, than from the pronunciations of other words in the same accent by means of G2P conversion. Hence, when a word's pronunciation is not available, as is fairly common in less prominent accents, it is better to first search for it in a different pronunciation dictionary, and if it is found, to convert it to the target accent. G2P methods should be reserved as a last resort. A summary of the accuracies for the different techniques is presented in Table 3.14.

Dictionary	Phoneme			Word		
	G2P	P2P	GP2P	G2P	P2P	GP2P
⇒ SAEDICT	90.87%	94.65%	96.14%	60.26%	72.37%	79.39%
⇒ BEEP	91.61%	94.92%	97.04%	64.55%	74.66%	84.87%
⇒ CMUDICT	91.33%	94.65%	97.01%	63.72%	73.69%	84.49%
⇒ PRONLEX	92.24%	94.85%	97.54%	66.16%	73.89%	86.58%
Average	91.51%	94.77%	96.93%	63.67%	73.65%	83.83%

Table 3.14: *Average accuracies achieved by G2P, P2P and GP2P conversion approaches.*

Converting between different English accents also gives considerably more accurate results than merely using another accent's dictionary, as can be seen from Table 3.4. Substituting one English dictionary for another with no modifications gives at best a 60% and at worst a 29% word accuracy. For P2P, however, even the poorest matching pair of dictionaries gives a conversion word accuracy above 72%. In addition, P2P conversion using automatically trained decision trees also performed considerably better than a system based on hand-crafted rules, as shown in Table 3.8.

The use of source graphemes in addition to source phonemes in GP2P conversion yielded a considerable further improvement in accuracy, as reflected in Table 3.14. On average, GP2P led to an absolute improvement in phoneme accuracy of 2.16% and in word accuracy of 10.18% relative to P2P. The accuracy improved for all dictionary pairs by an amount greater than the 95% confidence interval.

It is interesting to note that, when using GP2P, converting from PRONLEX to SAEDICT is almost as accurate as converting from BEEP to SAEDICT, despite there being a considerable difference both when comparing these dictionaries directly and when using P2P conversion.

GP2P also lead to a considerable reduction in the size of the decision trees, from an average of 3 840 nodes for P2P, to 2 541 for GP2P, a decrease of 34%. However, both P2P and GP2P trees are much smaller than G2P trees, which had 15 313 nodes on average. This reduction is to a large extent due to the use of pruning, which is beneficial for P2P and

GP2P but not G2P conversion. Even if no pruning is used, however, there is a considerable reduction in tree size, with P2P and GP2P having average tree sizes of 9 119 and 7 126 nodes respectively.

3.9 Chapter summary

This chapter has investigated methods that can be used to derive the pronunciation of a word in a new accent from its known pronunciation in another accent of the same language.

The performance of these techniques has been evaluated by applying them to three accents of English: RP, GenAm and SSAE. In order to achieve this, four pronunciation dictionaries were employed: BEEP for RP, CMUDICT and PRONLEX for GenAm, and SAEDICT for SSAE.

As a baseline for comparison, G2P techniques were first applied to the dictionaries individually, with comparable results in all four cases. The G2P algorithm was then modified to allow conversion of pronunciations from one accent to another, first using only the phonemes of the source accent (P2P) and then using both the phonemes and the graphemes (GP2P).

If we consider the specific case of SSAE, for which pronunciation resources are limited, our experiments show that the pronunciations of new words can be obtained with 82% accuracy using the developed GP2P conversion technique, in comparison with 60% that would be obtained using conventional G2P conversion.

Chapter 4

The effect of training sets on decision tree performance

The previous chapters have considered three techniques by means of which pronunciations for words not present in an existing target dictionary can be derived. All three techniques were tested in situations where training data were relatively abundant, totalling many thousands of words. This chapter will investigate the performance of these techniques when training data are scarce.

Firstly, the dependence of each technique's performance on the number of words in the training set will be considered. Subsequently, the effect of particular words in extremely small training sets will be investigated. The aim is to determine the optimal means of compiling a training set for a new accent of English, for which no pronunciations at all are yet available. Almost all the English accents used in South Africa, for example, fall into this category. The effectiveness of small training sets compiled in various ways will be compared with the use of well-established phonetically rich English word lists, such as the Wells keywords [49] and the SCRIBE corpus [37].

The experiments in this chapter will focus specifically on the determination of SSAE pronunciations. The BEEP dictionary will be employed for the application of the P2P and GP2P techniques. Other accent combinations will not be considered, although it is hoped that the results obtained for SSAE will be generally applicable to English accents.

4.1 Incremental training of decision trees

Possible modifications to the decision tree training algorithm that would allow trees to be trained incrementally were investigated. Using the algorithm discussed in Section 2.2, an entire tree must be trained at once, using all available training data. There are, however, numerous instances where it is advantageous to be able to rather modify an existing tree using some newly acquired portion of the training data, such that the resulting tree is identical to the tree that would have resulted from training a new tree using all the data. The tests in

this chapter, which consider the accuracy of a decision tree trained with increasing training data, could potentially benefit from incremental training.

Utgoff et al have developed such an incremental algorithm [44]. When a new training instance is to be incorporated into the tree, it is added to each node along the path determined by the existing tree's questions. All the nodes visited along this path are marked to indicate that they might change - unmarked nodes need not be investigated further. Marked nodes are then recursively updated, starting at the root and working towards the leaves, to ensure they still have the optimal question based on the training data associated with them.

To allow the question associated with a node to be changed without the need to retrain its entire subtree, an operation known as *transposition* was developed [44]. This operation, illustrated in Figure 4.1, is based on the fact that if a node's two children have the same question, it is possible to exchange the node's question with that of its children. This changes the position of two of the node's grandchildren, as the True-False and False-True grandchildren exchange places. There is, however, no change to any of the grandchildren or their subtrees. This operation therefore allows a question to be moved higher up the tree without significantly affecting the tree below.

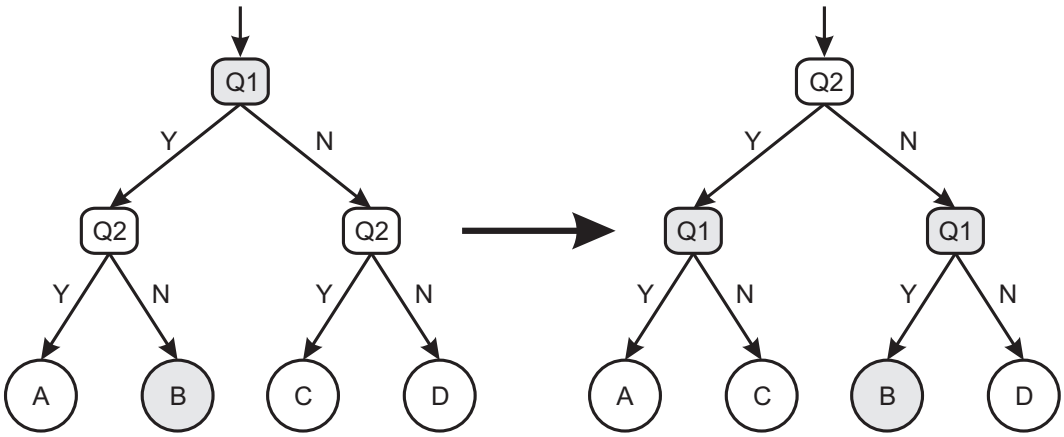


Figure 4.1: *The transposition operator for moving questions within a decision tree.*

By repeatedly applying the transposition operator, it is possible to move a question up to any node in the decision tree. This allows those nodes whose question is no longer optimal after a new training instance has been added, to be assigned a new question without completely retraining the tree.

When decision trees are applied to G2P, P2P or GP2P conversion, the most expensive process during training is calculating which potential question gives the greatest entropy gain. This is both due to the large number of potential questions, and also the relatively large number of classes (output phonemes) involved. Because adding one word adds a number of training instances to the tree (one per grapheme for G2P conversion, and one per source phoneme for P2P and GP2P conversion), the incremental training algorithm still

needs to check the questions associated with quite a number of nodes. For all these nodes, this expensive entropy-gain calculation must be performed. For this reason the incremental algorithm did not substantially influence the speed with which trees were trained.

While the incremental algorithm was implemented and preliminary tests were run, it was decided not to continue with its use as it gives no major benefit while being considerably more complex than standard decision tree training, and requiring additional code that needs maintenance.

4.2 Experimental setup

As stated at the start of this chapter, the aim is to determine the relationship between conversion accuracy and training dictionary size for the G2P, P2P and GP2P algorithms. This was achieved by running tests using increasing portions of the available training data, and determining the resulting conversion accuracy. In order to ensure that the results for the different tests can be properly compared, a large portion of the total data was reserved for testing. Of the 24 994 words common to both SAEDICT and BEEP, 10 000 were reserved for testing and the remainder for training. Hence all experiments were performed using this same 10 000 word test set.

To produce a fair split between training and testing data, the total available data were first sorted alphabetically, after which groups of 150 and 100 words at a time were assigned to the training and testing sets respectively. Words in the Wells and SCRIBE sets (described in Section 4.4) were included in the training data before this split was performed, to guarantee that they never formed part of the test set.

4.3 The effect of training set size

The first experiments were performed to determine the relationship between training set size and pronunciation accuracy. This was done for all three conversion techniques: G2P, P2P and GP2P.

The training sets used in these experiments were formed by choosing words at random from the 14994 available words of training data. For training set sizes of under 1000 words, 1000 random selections were chosen and decision trees trained for each. This allowed mean performance as well as 95% intervals to be determined. For training set sizes of more than 1000 words, only 100 random selections were used, to keep computation feasible.

For each training set, trees were trained and tested using increasing numbers of words. To make computation feasible, and also because the pronunciation accuracies quickly reach a near-steady state, the increments at which results were obtained were chosen to be further apart for larger numbers of words. Table 4.1 shows the increments used.

Number of words	Increment
1 - 25	1
25 - 150	25
150 - 500	50
500 - 1000	100
1000 - 2000	250
2000 - 5000	500
5000 - 14994	1000

Table 4.1: *Training set increments at which accuracies were determined.*

Table 4.2 gives the accuracy achieved when using the full 14 994 words of training data. These results are similar to those listed in Tables 3.5, 3.9 and 3.13, which were obtained using 10 times cross validation and somewhat larger training sets.

	Phoneme	Word	Size
G2P	89.17%	52.38%	14 063
P2P	94.22%	70.54%	7 213
GP2P	95.34%	75.39%	6 735

Table 4.2: *Accuracies of G2P, P2P and GP2P respectively, using all 14994 words in the training set.*

Figures 4.2, 4.3, 4.4 and 4.5 show how the performance changes with increasing training set size. GP2P values are indicated in blue, with the darkly shaded area indicating the 95% confidence interval around the mean, and the lightly shaded area showing the best and worst results obtained among the random samples. Mean values for G2P and P2P are indicated in green and red respectively.

Figures 4.2 and 4.4 show the detailed behaviour for the first 1000 words, while Figures 4.3 and 4.5 show the accuracy for all training set sizes up to the maximum of 14 994 words.

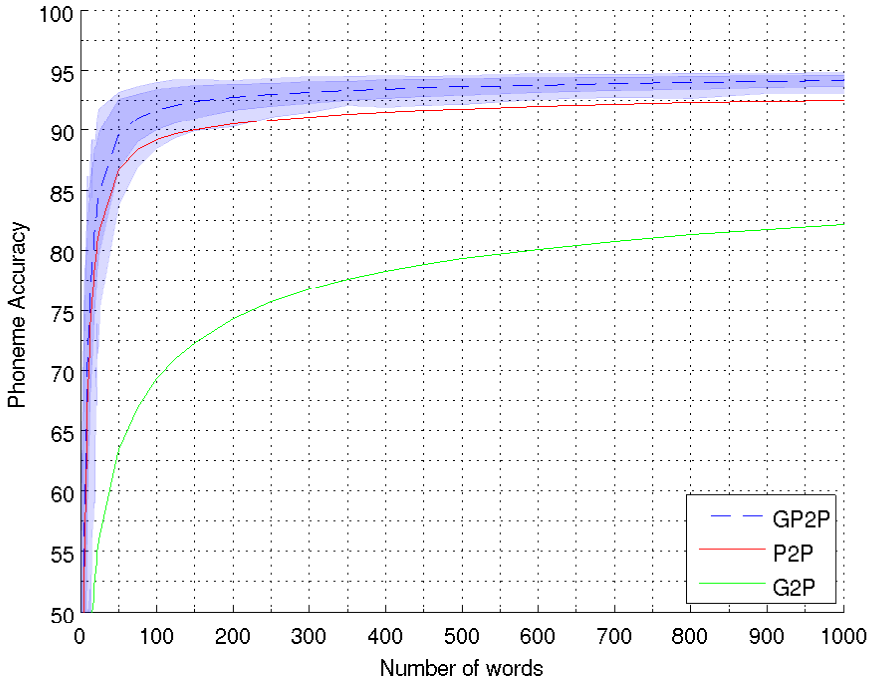


Figure 4.2: *Phoneme accuracy for training sets containing up to 1000 words.*

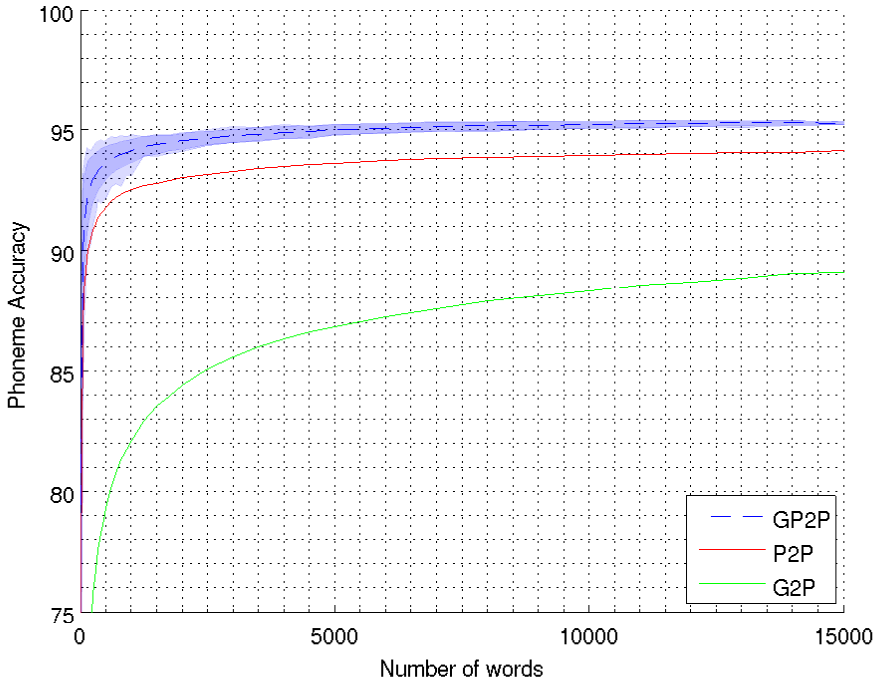


Figure 4.3: *Phoneme accuracy for training sets up to the maximum size of 14994 words.*

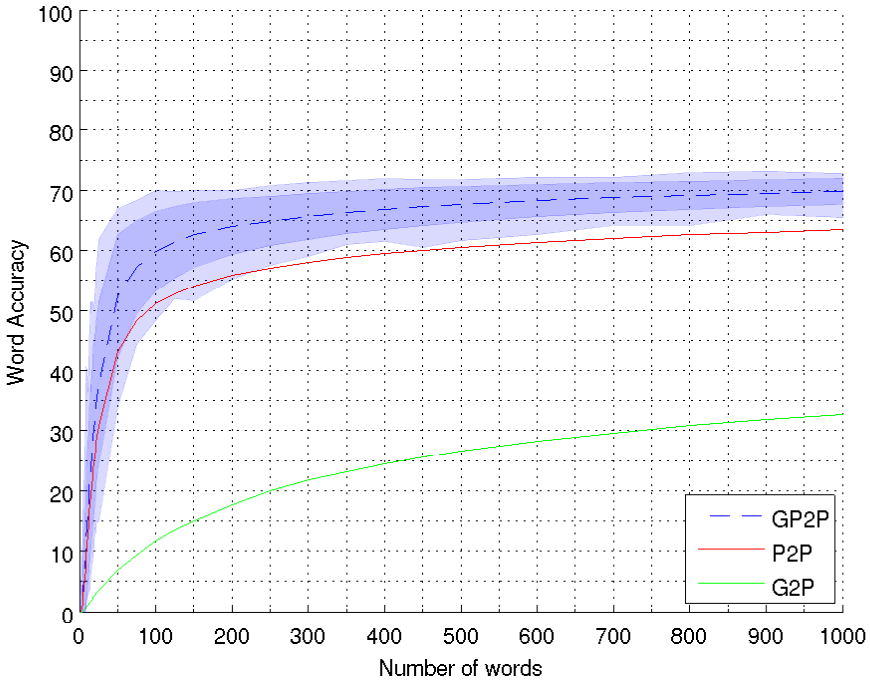


Figure 4.4: *Word accuracy for training sets containing up to 1000 words.*

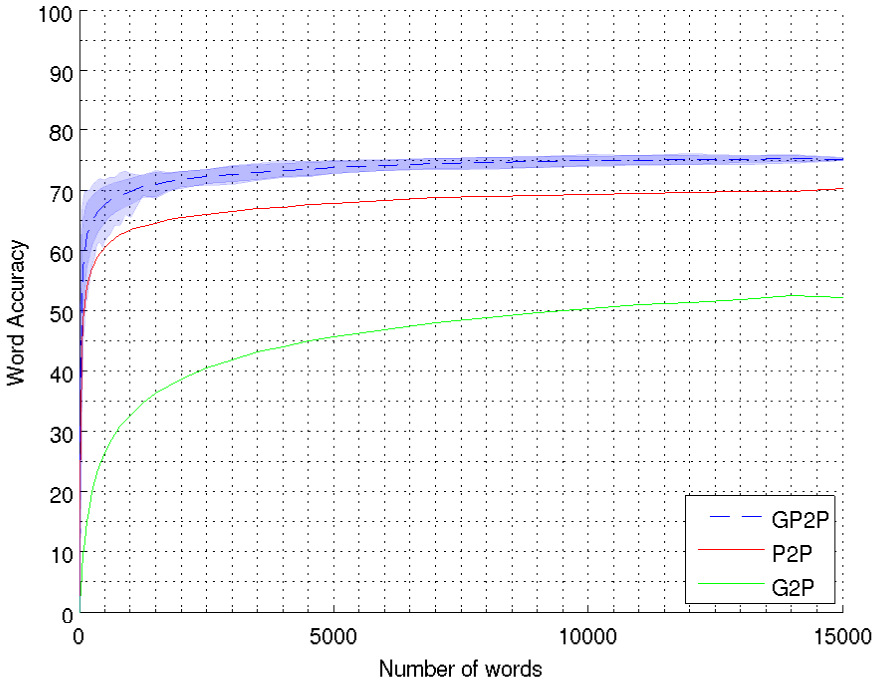


Figure 4.5: *Word accuracy for training sets up to the maximum size of 14994 words.*

Figure 4.6 shows how the trees' size increases as more training data is used. It is clear that in all three cases the tree size increases almost linearly with increasing training size. It can further be observed that the relationship between the different technique's tree sizes observed in Chapter 3 holds for any size of training set.

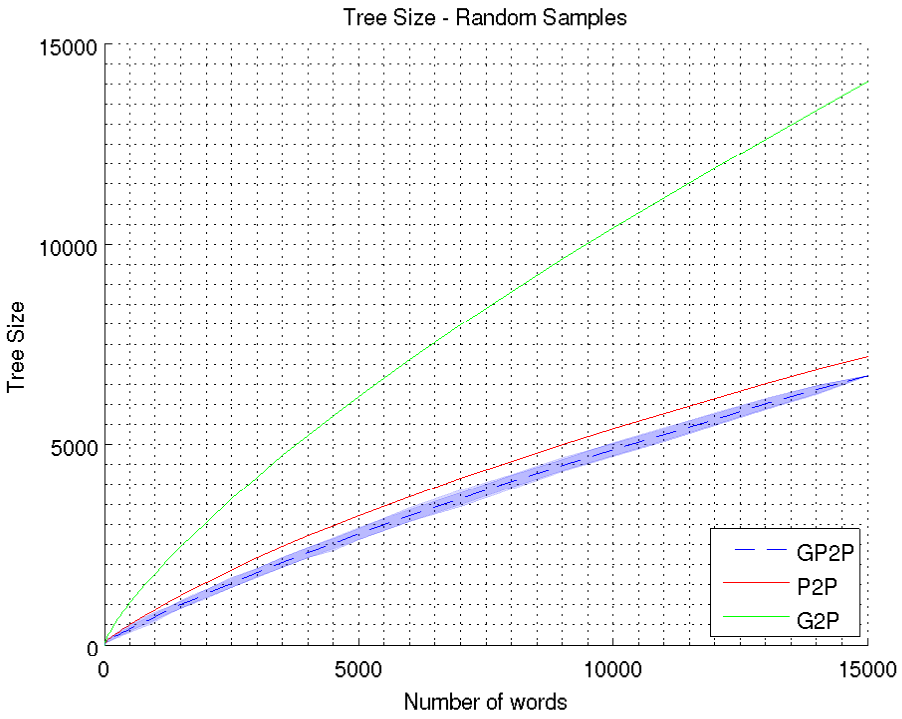


Figure 4.6: *Tree size for different training set sizes.*

From these figures the following observations can be made and conclusions drawn:

- The performance of GP2P is better than the performance of P2P for all training set sizes.
- The performance of P2P is better than the performance of G2P for all training set sizes.
- The knee of the GP2P accuracy curve is substantially sharper than that of the P2P curve, and both are much sharper than the knee of the G2P curve.
- For small training set sizes (fewer than 200 words), the particular words in the training set have a substantial effect on GP2P performance. For example, for a training set size of 100 words, the 95% intervals are 3.38% for phoneme and 13.16% for word accuracy respectively, while at 1000 words the corresponding figures are 0.95% and 4.29%.
- GP2P and P2P conversion make use of substantially smaller trees than G2P, irrespective of the training set size.

Finally, and very significantly, it is clear from Figures 4.2 and 4.4 that GP2P conversion can deliver near-optimal performance for remarkably small training sets.

Number of words	Average results		Best results	
	Phoneme	Word	Phoneme	Word
100	96.19%	79.43%	98.62%	92.90%
200	97.30%	84.94%	98.75%	92.92%
300	97.74%	87.21%	99.08%	94.65%
400	98.02%	88.71%	99.21%	95.49%
500	98.22%	89.79%	99.17%	95.22%
600	98.37%	90.61%	99.36%	95.81%
700	98.49%	91.25%	99.34%	95.81%
800	98.58%	91.80%	99.40%	96.75%
900	98.67%	92.27%	99.37%	97.10%
1000	98.74%	92.68%	99.43%	96.58%

Table 4.3: *The accuracy of GP2P conversion, given as a percentage of the accuracy obtained when including the entire training set of 14994 words.*

Table 4.3 shows how close the average and the best-performing GP2P systems depicted in Figures 4.2 and 4.4 are to the accuracy achieved when including the entire training set. It is evident that a system whose phone accuracy is within 99% of the optimum can be achieved with as few as 300 words.

4.4 The choice of training set words

The experiments in the previous section clearly showed that

- High accuracies can be achieved wth GP2P conversion using fairly small training sets.
- For small training sets, the particular words in the set considerably affect performance.

Hence, the objective of this section is to investigate the effect of the choice of training set words when this set is small. Specifically, we try to determine an optimal strategy for choosing a small training set, and compare it with some well-known standards, such as the Wells keywords.

A number of specific strategies of compiling the training set were tested to determine their effect on the system's performance. Random selection, as used in the previous section, was used as a baseline. The strategies considered were:

1. Using the most frequent words first
2. Using the longest words first
3. Using the shortest words first
4. Using words with the greatest number of different graphemes first
5. Using words with the greatest number of different phonemes first
6. Using words drawn from the SCRIBE sentences
7. Wells keywords [49] supplemented by random selections.

The SCRIBE corpus [37] defines two sets of sentences: 200 'phonetically rich' sentences (Set A) and 460 'phonetically compact' sentences (Set B). The words from these sentences were considered as candidates for composing the GP2P training sets. Set A and B yielded 1071 and 1523 words respectively that are present in both dictionaries.

Figures 4.7, 4.8, 4.9 and 4.10 show the performance attained using the most frequent words, the longest words, and the SCRIBE A and B words respectively. The SCRIBE results are only shown for the first 1000 and 1500 words respectively, and indicate three alternatives: the mean performance, taken over 1000 random samples, the best of these random orderings, and also an optimal ordering of the words. This optimal ordering is determined using the method described in Section 4.4.1.

The other strategies tested (shortest words, Wells keywords and the words with the greatest number of different graphemes or phonemes) performed poorly in comparison with randomly chosen words, and consequently these results are not shown.

From Figures 4.7 and 4.8 it is clear that, while selection based on the frequency or the length of words is not a bad criterion, it is almost always outperformed by random selections.

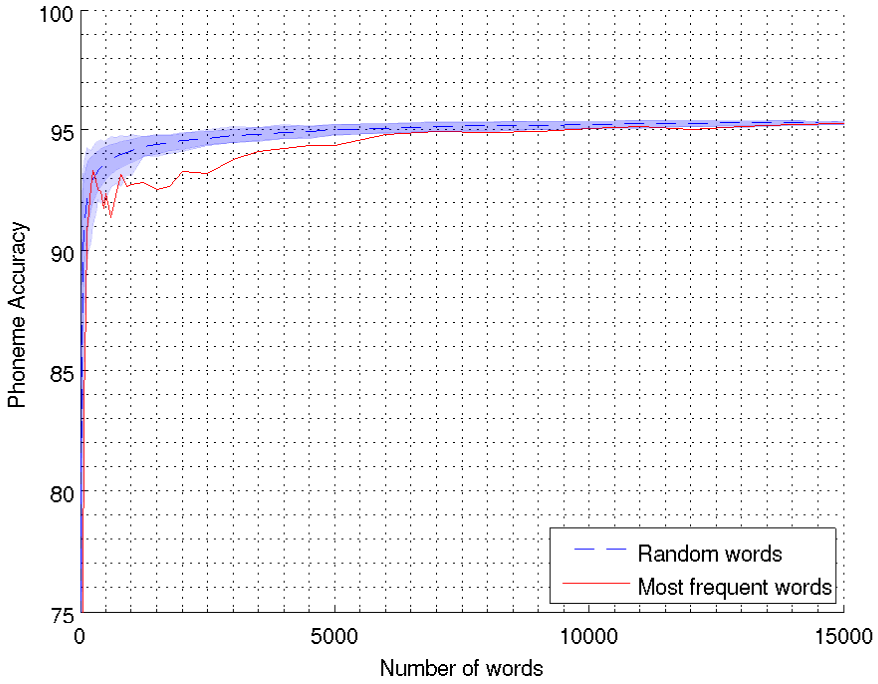


Figure 4.7: *Phoneme accuracy when composing the training set of the most frequent words.*

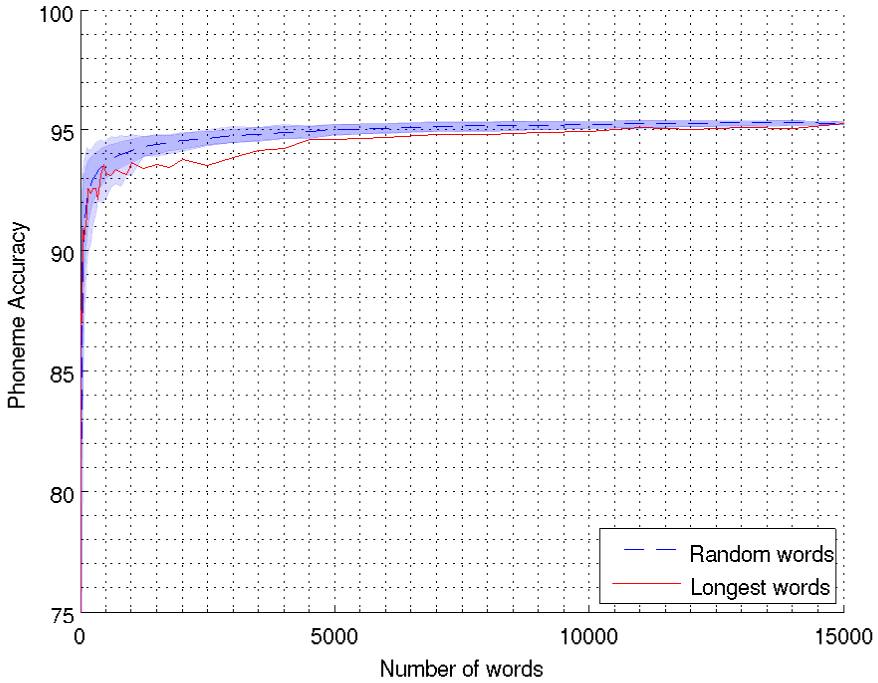


Figure 4.8: *Phoneme accuracy when composing the training set of the longest words.*

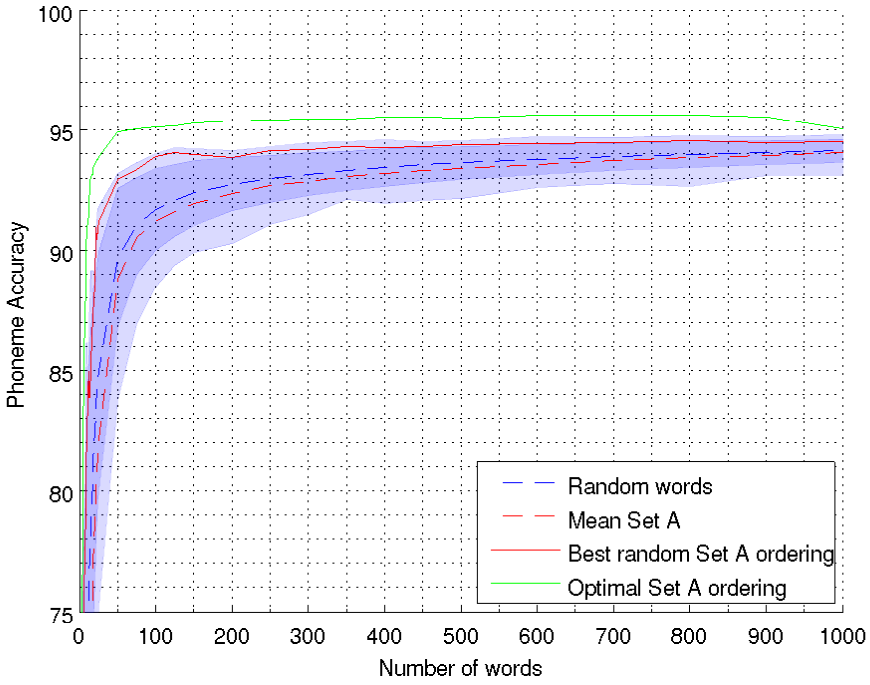


Figure 4.9: *Phoneme accuracy when composing the training set of SCRIBE Set A (phonetically rich) words.*

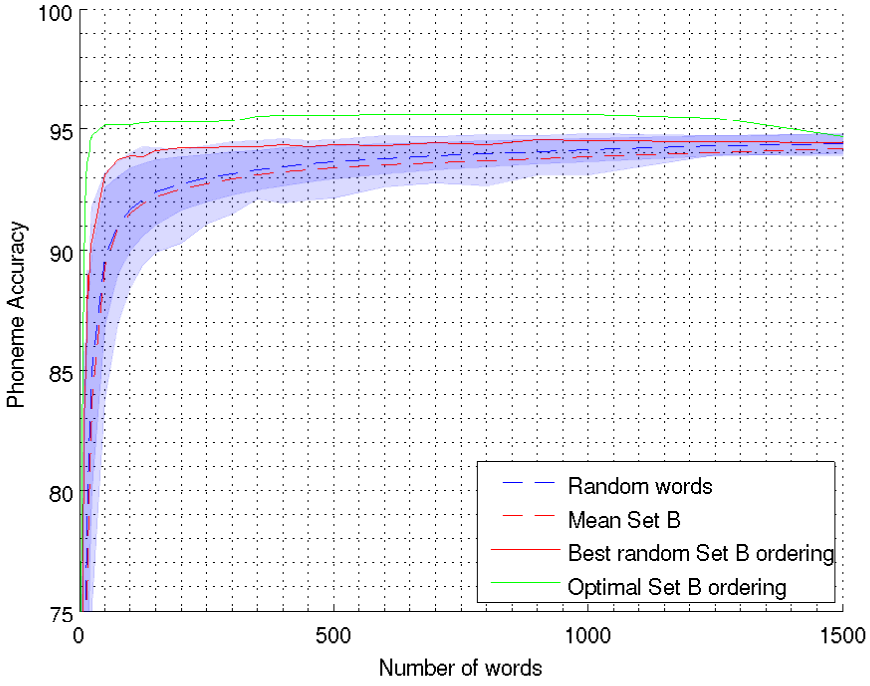


Figure 4.10: *Phoneme accuracy when composing the training set of SCRIBE Set B (phonetically compact) words.*

Figures 4.9 and 4.10, on the other hand, show that the selection of training set words from the SCRIBE sentences is a very good strategy, which in the optimal case can clearly outperform random selections.

4.4.1 Optimal words

A final approach was to search all the available words in the training set and determine a list of words that gave optimal GP2P conversion performance. The words were chosen sequentially by means of a greedy algorithm, whereby each word was chosen so as to yield the maximum increase in performance above that of the existing list of previously chosen words. The first 60 of these words are listed in Appendix E.

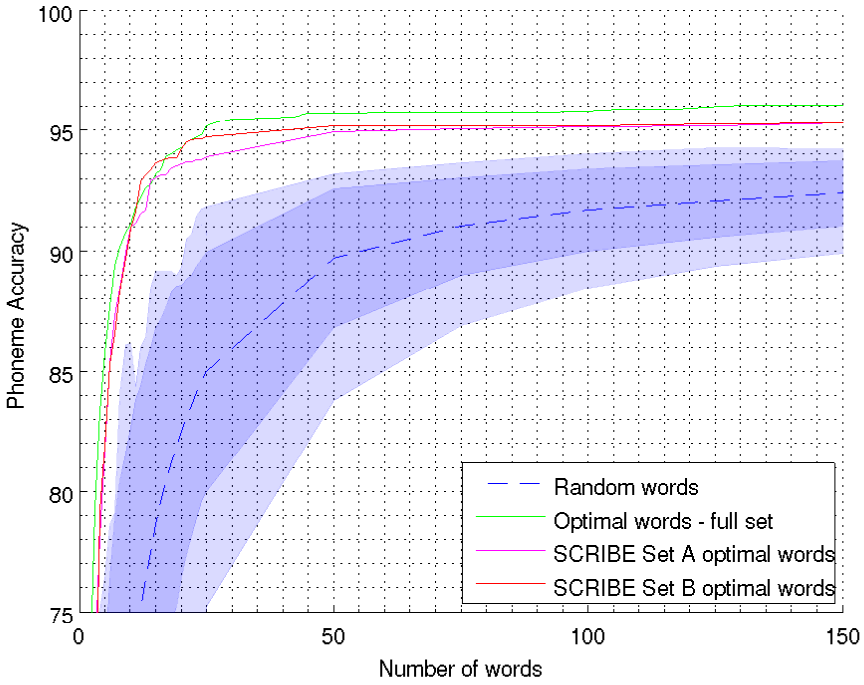


Figure 4.11: *Phoneme accuracy achieved using an optimally determined training set.*

Figure 4.11 gives the phoneme accuracy for this optimal list, for the first 150 words. This figure shows that training set words chosen in this way can improve on the phone accuracy obtained by the best SCRIBE-based selection by approximately 1%.

4.5 Chapter Summary and conclusion

When comparing the values in Table 4.2 and Figures 4.3 and 4.5, it is clear that not only does GP2P perform considerably better than both P2P and G2P, but also that GP2P results improve more rapidly with increasing training data than P2P, and both GP2P and P2P improve much faster than G2P.

Considering Table 3.4, a GP2P system needs to achieve a phoneme accuracy higher than 92.1% to improve on the unmodified source dictionary. From Figure 4.3 it can be seen that on average 100 randomly chosen words are needed to achieve this, and in the worst case approximately 500 words. While GP2P conversion performance continues to improve as data is added, the rate of improvement decreases sharply after the first 500 words, and after 3000 words there is almost no further improvement.

Of the specific lists used, only the SCRIBE sets compare favourably to randomly chosen words. For a random ordering the SCRIBE sets do almost as well as randomly chosen words, and for an optimal arrangement they perform almost as well as an optimal list drawn from the training set.

The optimally chosen word list achieves a performance far better than any other. After a mere 9 words the accuracy is higher than the dictionary's existing 92.1% correspondence, and 25 words already give performance within 1% of the accuracy achieved using the full 14 994 words of training data. After 50 words the performance increases very slowly.

It still remains to be established whether the optimal training sets described in this chapter are also optimal for accent pairs other than RP / SSAE. However, the results of this chapter do appear to indicate that it is possible to train almost optimally-performing GP2P systems using extremely small training sets. This is a highly significant result for the development of pronunciation dictionaries for under-developed or under-resourced accents.

Chapter 5

Software implementation

The G2P, P2P and GP2P conversion systems comprise a number of largely disjoint tasks, notably loading the dictionary and converting it to a useable format, aligning the dictionary entries, building grapheme and phoneme clusters, generating a list of all possible questions, building the decision tree and finally running tests on the decision tree. The system was implemented so as to reflect these different steps, with a separate module for each primary task. To prevent the need to maintain multiple versions of the same code, the system was written so that only parameters need to be changed to determine which of the three conversion techniques (G2P, P2P or GP2P) is used. All the software was written in C, with some additional scripting done in Python.

5.1 Symbols and Dictionary

The first two modules, `symbols` and `dictionary`, provide basic functionality needed by other modules.

`Symbols` defines the possible symbols (both graphemes and phonemes) in use. Since a number of the algorithms operate identically on both graphemes and phonemes, especially when searching for and selecting questions while building the tree, a single enumerated data type is used for both. `Symbols` contains functions to convert graphemes and phonemes from a text-based, human-readable representation to the internal representation used within the program and vice-versa. Finally there are functions for combining and splitting phonemes, necessary to create pseudophonemes so as to emulate one-to-one grapheme-phoneme alignments when a single grapheme aligns with multiple phonemes.

`Dictionary` provides basic functions to manipulate the pronunciation dictionary, particularly loading it from an external file, splitting it into training and testing dictionaries and comparing two dictionaries. Dictionaries are stored as linked lists of words, with each node containing arrays of symbols for both the spelling and the pronunciation of the word.

5.2 Alignment

The `alignment` module aligns graphemes and phonemes for all words in a dictionary so that they can be used to train the system. This is done according to the method described in Section 2.1. Various different initalisation methods are possible. The dictionary is aligned by inserting nulls into the phoneme string, and also combining phonemes into double phonemes where necessary. It is possible to either allow any possible grapheme-phoneme mappings, or to limit them to a hand-defined list of possible matches.

The module also allows the alignment of groups of graphemes to single phonemes, for use in GP2P conversion.

5.3 Clustering

The `cluster` module builds cluster trees of graphemes or phonemes, grouping them together to minimise the information entropy of the corresponding phonemes in the training dictionary. This is done according to the algorithm described in Section 2.3.1. Separate cluster trees are built for each context position, i.e. different trees for the focus grapheme, the grapheme to its left, the grapheme to its right, the latest phone derived, and so forth. Using multiple trees in this way more accurately reflects the influence that categories of symbols have on the output phoneme.

There is a function to "flatten" a tree, reducing it to a linear linked list by means of a breadth-first search. This makes it easier to traverse all the clusters when searching for questions. For easy retrieval, the clusters themselves are stored as a list of boolean flags indicating whether the corresponding symbol is in the cluster or not.

The clusters can be limited so that further context positions only contain larger clusters, an approach suggested in [22]. This is intended to prevent questions becoming over-trained, and forces questions regarding distant symbols to be less specific.

5.4 Questions

The `question` module contains all functionality relating to decision tree questions. It firstly allows a list of potential questions to be generated, based on the current system parameters. There are then functions to calculate the resulting entropy if a given set of context data are split according to a particular question.

5.5 Decision Trees

The `dectree` module provides functions that build and query decision trees. Before these are used, there is a function that takes the dictionary and builds separate context strings

for each position in each word. This allows these to be used separately during training. There is also a function that creates a summary of this data, which is useful for calculating the entropy gain of different questions. The most important function in this module is the one that builds the decision tree, according to the algorithm described in Section 2.2. To make this easier, a number of internal functions exist for performing tasks like calculating the entropy gain for a given question and splitting the data at a node once a question has been chosen. Once the tree has been built, there is a function that prunes the tree using a separate, pruning dataset.

5.6 Parameters

In order to facilitate testing, especially of different combinations of parameters, the `parameters` module has functions that can set the global parameters for the whole system. These can be read in from an external file, or can take some default value. These default values are contained in the `constants.h` file. This module will also set certain parameters automatically depending on whether the system is operating in G2P, P2P or GP2P mode.

5.7 Testing

The `testing` module contains all the functions to test a decision tree, once it has been trained. For this the tree must first be queried to obtain pronunciations for the test data. These pronunciations are then aligned to the canonical ones, and phoneme and word accuracies determined from these alignments. There are also functions to determine a confidence interval for the phoneme accuracy, and the POI for those cases where there exists a baseline tree for comparison.

5.8 Chapter Summary

This chapter has given a broad overview of the system implementation and functioning. Further detail can be obtained from the well-commented source code.

Chapter 6

Summary and conclusions

This thesis has investigated various data-driven techniques by which pronunciation dictionaries can be automatically augmented. First, grapheme-to-phoneme (G2P) conversion techniques used by other researchers were evaluated, and decision trees were chosen as a basis for further work. Decision tree based G2P conversion was then evaluated for Standard South African English (SSAE), British English (RP) and American English (GenAm) by means of four appropriate dictionaries: SAEDICT, BEEP, CMUDICT and PRONLEX.

Building on this work, the decision tree algorithm was extended to allow the conversion of pronunciations between different pronunciation dictionaries by means of phoneme-to-phoneme (P2P) and grapheme-and-phoneme-to-phoneme (GP2P) conversion.

Finally, the three techniques (G2P, P2P and GP2P conversion) were compared in terms of their performance for various training set sizes.

6.1 G2P conversion

A thorough investigation was made of decision tree based G2P conversion. Extensive tests showed that the parameters for which the decision trees gave the best G2P conversion included a context window of three graphemes to either side of the focus grapheme, as well as the three previously generated phonemes and null questions relating to these phonemes. Clusters were used to obtain group questions. Pruning the tree did not improve performance. This optimal configuration agrees with that employed by other authors [22, 48].

The performance of our G2P implementation compares well with those found in the literature, most of which give phoneme accuracies of between 89% and 91% [9, 39, 33].

For the most part, varying the parameters of the system had little effect on its accuracy. This suggests that merely adding questions, or changing the parameters with which the trees are trained is unlikely to lead to any substantial further G2P performance improvement.

6.2 P2P conversion

The decision trees developed for G2P conversion were modified to allow the conversion of pronunciations between different dictionaries by means of P2P conversion. During P2P conversion the phonemes of the source accent are used as input to the decision trees, rather than graphemes, as is the case for G2P conversion.

P2P conversion was evaluated using the four dictionaries used during G2P experiments. It was found that, when the pronunciation is needed for a word not present in the target accent, it is substantially more accurate to modify an existing pronunciation of the word in a different accent, by means of P2P conversion, than to derive it solely from the word's spelling, using G2P conversion.

This in itself is an important result for the development of South African pronunciation dictionaries, and has been published in the referred proceedings of an international conference [26].

6.3 GP2P conversion

Building on the successes of P2P conversion, GP2P conversion aimed to achieve further improvements by incorporating the graphemes, as well as the phonemes of the source accent into the decision tree input. The P2P conversion algorithm was thus modified to allow the addition of the source orthography, suitably aligned with the source phonemes as sequences of zero or more graphemes, to be added to the context used when determining decision tree questions.

GP2P conversion was shown to lead to additional performance improvements relative to P2P for all four dictionaries.

6.4 Training set size and composition

Finally, experiments were performed to determine how large a training dictionary in the target accent must be available to accurately obtain pronunciations. To do this the G2P, P2P and GP2P conversion accuracies were determined as a function of varying quantities of training data.

It was found that GP2P and P2P conversion require a substantially smaller training set than G2P conversion. Furthermore, it was established that very little training data is needed to train a decision tree to perform GP2P conversion at almost maximum accuracy. The bulk of the system's accuracy is achieved within the first 500 words. After 3000 words, there is almost no further improvement.

For the best randomly selected words observed, a mere 300 words led to an accuracy that was within 1% of that achieved using the entire training set. Several specific approaches to

compiling the training set for GP2P conversion were tested. Only the SCRIBE sets performed at a level comparable with randomly selected words, however. An optimal set of training words was determined by means of a greedy algorithm. Using this optimal set, a mere 25 words already achieved an accuracy within 1% of that of the entire training data.

6.5 Further work

There are several avenues of further research using the techniques considered and developed in this thesis:

- The accent conversion methods can be tested by their inclusion in a TTS and/or ASR system. It can then be determined whether improved user ratings in listening tests (for TTS) or improved error rates (for ASR) are achieved relative to a G2P baseline.
- The accent conversion techniques, in conjunction with the findings regarding training set sizes, can be applied to other South African English accents, such as Indian, Coloured and Black English. These accents are currently severely under-resourced.
- It can be explored whether the findings for SSAE, RP and GenAm also hold for other varieties of English, such as Hong-Kong and Australian English. The techniques can also be applied to other languages with accent variation, such as French and Portuguese, or to closely related languages like Dutch and Afrikaans, or isiXhosa and isiZulu.

6.6 Conclusion

This thesis has investigated three automatic, data-driven techniques by which pronunciations may be generated for words not present in a pronunciation dictionary. The first method, G2P conversion, is widely used and the results obtained agree with those of other researchers. The other two, P2P and GP2P conversion, are novel methods of converting pronunciations between different accents. Both P2P and GP2P conversion were found to substantially outperform G2P conversion, and also to require much smaller training data sets.

Pronunciation dictionaries form an essential part of many aspects of speech technology. The work developed in this thesis can provide a method by which such dictionaries can be effectively and efficiently created for under-resourced accents, potentially granting speakers of those accents access to a much wider spectrum of speech-based technology.

Bibliography

[1] Adamson, M. and Damper, R., "A recurrent network that learns to pronounce English text." in *Proc. ICSLP*, (Philadelphia, USA), pp. 1704–1707, 1996.

[2] Andersen, O., Kuhn, R., Lazaridès, A., Dalsgaard, P., Haas, J., and Nöth, E., "Comparison of two tree-structured approaches for grapheme-to-phoneme conversion." in *Proc. ICSLP*, (Philadelphia, USA), 1996.

[3] Bakiri, G. and Dieterrich, T., "Performance comparison between human engineered & machine learned letter-to-sound rules for English: a machine learning success story." in *Proc. International Conference on the applications of Computer and Statistics to Science and Society*, (Cairo, Egypt), 1993.

[4] BEEP, "The British English Example Pronunciation (BEEP) Dictionary, http://svr-www.eng.cam.ac.uk/comp.speech/Section1/Lexical/beep.html." 2009. (accessed March 2009).

[5] Bekker, I., *The vowels of South African English*. Potchefstroom, South Africa: PhD Thesis, North-West University, 2009.

[6] Bellagarda, J., "Unsupervised, language-independent grapheme-to-phoneme conversion by latent analogy." *Speech Communication*, 2005, Vol. 46, No. 2, pp. 140–152.

[7] Bisani, M. and Ney, H., "Bootstrap estimates for confidence intervals in ASR performance evaluation." in *Proc. ICASSP*, (Montreal, Canada), 2004.

[8] Bisani, M. and Ney, H., "Joint-sequence models for grapheme-to-phoneme conversion." *Speech Communication*, 2008, Vol. 50, No. 5, pp. 434–451.

[9] Black, A., Lenzo, K., and Pagel, V., "Issues in building general letter to sound rules." in *Proc. ESCA Speech Synthesis Workshop*, (Jenolan Caves, Australia), 1998.

[10] Bowerman, S., "White South African English: phonology." in *A Handbook of varieties of English* (Schneider, E., Burridge, K., Kortmann, B., Mesthrie, R., and Upton, C. (Eds)), vol. 1, pp. 931 – 942, Mouton de Gruyter, Berlin, 2004.

[11] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J., *Classification and regression trees*. Pacific Grove: Wadsworth & Brooks, 1984.

[12] Chen, S., "Conditional and joint models for grapheme-to-phoneme conversion." in *Proc. Eurospeech*, (Geneva, Switzerland), pp. 2033–2036, 2003.

[13] CMU, "Carnegie Mellon University Pronouncing Dictionary, http://www.speech.cs.cmu.edu/cgi-bin/cmudict." 2009. (accessed March 2009).

[14] Daelemans, W. and van den Bosch, A., "Language-independent data-oriented grapheme-to-phoneme conversion." in *Progress in Speech Synthesis* (van Santen, J., Sproat, R., Olive, J., and Hirschberg, J. (Eds)), pp. 77–89, Springer, New York, 1994.

[15] Damper, R., Marchand, Y., Adamson, M., and Gustafson, K., "Comparative evaluation of letter-to-sound conversion techniques for English text-to-speech synthesis." in *Proc. ESCA/COCOSDA International Workshop on Speech Synthesis*, (Jenolan Caves, Australia), pp. 53–58, 1998.

[16] Damper, R., Marchand, Y., Adamson, M., and Gustafson, K., "Evaluating the pronunciation component of text-to-speech systems for English: a performance comparison of different approaches." *Computer Speech and Language*, 1999, Vol. 13, No. 2, pp. 155–176.

[17] Demberg, V., Schmid, H., and Möhler, G., "Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion." in *Proc. ACL*, (Prague, Czech Republic), 2007.

[18] Galescu, L. and Allen, J., "Bi-directional conversion between graphemes and phonemes using a joint n-gram model." in *Proc. ISCA ITRW on Speech Synthesis*, (Perthshire, Scotland), 2001.

[19] Huang, J., Olorenshaw, L., Hernández-Ábrego, G., and Duan, L., "Memory efficient grapheme-to-phoneme conversion system for speech processing." in *Proc. ICSLP*, (Jeju, Korea), pp. 1237–1240, 2004.

[20] Humphries, J., Woodland, P., and Pearce, D., "Using accent-specific pronunciation modelling for robust speech recognition." in *Proc. ICASSP*, (Salt Lake City), 2001.

[21] Jensen, K. and Riis, S., "Self-organizing letter code-book for test-to-phoneme neural network model." in *Proc. ICSLP*, (Beijing, China), pp. 318–321, 2000.

[22] Kienappel, A. K. and Kneser, R., "Designing very compact decision trees for grapheme-to-phoneme transcription." in *Proc. Eurospeech*, (Aalborg, Denmark), pp. 1911–1914, 2001.

[23] Ladefoged, P., *Vowels and consonants, second edition*. Massachusetts, USA: Blackwell Publishing, 2005.

[24] LDC, "COMLEX English pronouncing lexicon, http://www.ldc.upenn.edu." 2009. (accessed March 2009).

[25] Loots, L., Davel, M., Barnard, E., and Niesler, T., "Comparing manually-developed and data-driven rules for P2P learning." in *Proc. PRASA*, (Stellenbosch, South Africa), 2009.

[26] Loots, L. and Niesler, T., "Data-driven phonetic comparison and conversion between South African, British and American English pronunciations." in *Proc. Interspeech*, (Brighton, UK), 2009.

[27] Marchand, Y. and Damper, R., "A multistrategy approach to improving pronunciation by analogy." *Computational Linguistics*, 2000, Vol. 26, No. 2, pp. 195–219.

[28] Meron, J. and Veprek, P., "Compression of exception lexicons for small footprint grapheme-to-phoneme conversion." in *Proc. ICASSP*, (Philadelphia, USA), pp. 293–296, 2005.

[29] Mimer, B., Stker, S., and Schultz, T., "Flexible decision trees for grapheme based speech recognition." in *Proc. Conference Elektronische Sprachsignalverarbeitung (ESSV)*, (Cottbus, Germany), 2004.

[30] Mitten, R., "Computer-useable version of the Oxford Advanced Learner's Dictionary of Current English, Tech. Rep., Oxford Text Archive." 1992.

[31] Niesler, T., Louw, P., and Roux, J., "Phonetic analysis of Afrikaans, English, Xhosa and Zulu using South African speech databases." *Southern African Linguistics and Applied Language Studies*, 2005, Vol. 23, No. 4, pp. 459–474.

[32] Pagel, V., Lenzo, K., and Black, A., "Letter to sound rules for accented lexicon compression." in *Proc. ICSLP*, (Sydney, Ausralia), 1998.

[33] Polyakova, T. and Bonafonte, A., "Learning from errors in grapheme-to-phoneme conversion." in *Proc. ICSLP*, (Pittsburgh, USA), pp. 2442–2445, 2006.

[34] Rabiner, L. and Juang, B., *Fundamentals of speech recognition*. New Jersey, USA: Prentice Hall, 1993.

[35] Reichel, U. and Pfitzinger, H. R., "Text preprocessing for speech synthesis." in *Proc. TC-STAR Workshop on Speech-to-Speech Translation*, (Barcelona, Spain), 2006.

[36] Rentzepopoulos, P. and Kokkinakis, G., "Efficient multilingual phoneme-to-grapheme conversion based on HMM." *Computational Linguistics*, 1996, Vol. 22, No. 3, pp. 351–376.

[37] SCRIBE, "Spoken Corpus of British English, http://www.phon.ucl.ac.uk/resource/scribe/." 2009. (accessed Sept 2009).

[38] Sejnowksi, T. and Rosenberg, C., "Parallel networks that learn to pronounce English text." *Complex Systems*, 1987, Vol. 1, No. 1, pp. 145–168.

[39] Suontausta, J. and Häkkinen, J., "Decision tree based text-to-phoneme mapping for speech recognition." in *Proc. ICSLP*, (Beijing, China), pp. 831–834, 2000.

[40] Taylor, P., "Hidden Markov models for grapheme to phoneme conversion." in *Proc. Interspeech*, (Lisbon, Portugal), pp. 1973–1976, 2005.

[41] Tian, J., Suontausta, J., and Häkkinen, J., "Weighted entropy training for decision tree based text-to-phoneme mapping." in *Proc. Eurospeech*, (Geneva, Switzerland), pp. 217–220, 2003.

[42] Torkkola, K., "An efficient way to learn grapheme-to-phoneme rules automatically." in *Proc. ICASSP*, (Minneapolis, USA), pp. 199–202, 1993.

[43] Uebler, U., "Grapheme-to-phoneme conversion using pseudo-morphological units." in *Proc. ICSLP*, (Denver, USA), pp. 101–104, 2002.

[44] Utgoff, P., Berkman, N., and Clouse, J., "Decision tree induction based on efficient tree restructuring." *Machine Learning*, 1997, Vol. 29, No. 1, pp. 5–44.

[45] van den Bosch, A. and Daelemans, W., "Data-oriented methods for grapheme-to-phoneme conversion." in *Proc. conference of European chapter of the Association for Computational Linguistics*, (Utrecht, the Netherlands), pp. 45–53, 1993.

[46] van der Lubbe, J., *Information theory*. Cambridge, UK: Cambridge University Press, 1997.

[47] Wan, V., Dines, J., El Hannani, A., and Hain, T., "Bob: a lexicon and pronunciation dictionary generator." in *Proc. Spoken Language Technology Workshop*, (Goa, India), 2008.

[48] Webster, G. and Braunschweiler, N., "An evaluation of non-standard features for grapheme-to-phoneme conversion." in *Proc. Interspeech*, (Brisbane, Australia), 2008.

[49] Wells, J., *Accents of English*. Cambridge: Cambridge University Press, 1982.

[50] Yvon, F., "Self-learning techniques for grapheme-to-phoneme conversion." in *Proc. Onomastica Research Colloquium*, (London, UK), 1994.

Appendix A

Alignment theory

To find the optimal alignment between phonemes and graphemes, it is necessary to find the phoneme sequence p^*, given an orthography g, that will the maximise the posterior probability, as given by the following equation:

$$p^* = \arg\max_p P(p|g)$$

Taking i as the position in the word, g_i as the grapheme in that position and p_i as the phoneme or pseudophoneme aligned to that grapheme in the phoneme sequence p, this becomes

$$p^* = \arg\max_p \prod_i P(p_i|g_i)$$

Taking $N(p_i, g_i)$ as the number of correspondences between phoneme p_i and grapheme g_i in the data being aligned, and $N(g_i)$ as the number of occurrences of grapheme g_i, the conditional probability $P(p_i|g_i)$ may be approximated by the relative frequency $N(p_i, g_i)/N(g_i)$.

$$\begin{aligned} p^* &= \arg\max_p \prod_i \frac{N(p_i, g_i)}{N(g_i)} \\ &= \arg\max_p \frac{\prod_i N(p_i, g_i)}{\prod_i N(g_i)} \end{aligned}$$

The denominator $\prod_i N(g_i)$ is constant for a given word, and may thus be omitted, leaving the following:

$$p^* = \arg\max_p \prod_i N(p_i, g_i)$$

Finally, the logarithm can be taken, since this is a strictly increasing function.

$$\begin{aligned} p^* &= \arg\max_p \left(\log \prod_i N(p_i, g_i) \right) \\ &= \arg\max_p \sum_i \log N(p_i, g_i) \end{aligned}$$

Hence the optimal alignment between the phoneme and grapheme strings can be found by determining the alignment which maximises the above equation. This can be found using Dynamic Time Warping (DTW), a dynamic programming algorithm [32].

Appendix B

ARPABET Phoneme set

Table B.1 gives the full ARPABET phoneme set used, with the corresponding symbols in ASTBET (in which SAEDICT was transcribed), and also example English words. Asterisks indicate cases where a phoneme is not present in one of these phoneme sets, and has been mapped to the closest phoneme.

Description	ARPABET	ASTBET	English
Stops			
Voiceless Bilabial Plosive	p	p	pit
Voiced Bilabial Plosive	b	b	baby
Voiceless Alveolar Plosive	t	t	total
Voiced Avleolar Plosive	d	d	death
Voiceless Velar Plosive	k	k	kick
Voiced Velar Plosive	g	g	gun
Fricatives			
Voiceless Labiodental Fricative	f	f	four
Voiced Labiodental Fricative	v	v	vat
Voiceless Dental Fricative	th	th	thing
Voiced Dental Fricative	dh	dh	this
Voiceless Alveolar Fricative	s	s	some
Voiced Alveolar Fricative	z	z	zero
Voiceless Post-Alveolar Fricative	sh	sh	shine
Voiced Post-Alveolar Fricative	zh	zh	genre
Voiceless Glottal Fricative	hh	h	hand
Affricates			
Voiceless Alveolar Affricate	ch	t_lnksh	change
Voiced Alveolar Affricate	jh	d_lnkzh	jar

Approximants			
Alveolar Approximant	r	rt	red
Alveolar Lateral Approximant	l	l	legs
Palatal Approximant	y	j	yes
Voiced Labio-Velar Approximant	w	w	west
Aspirated Labio-Velar Approximant	wh	*w	white
Nasals			
Bilabial Nasal	m	m	man
Alveolar Nasal	n	n	not
Velar Nasal	ng	nj	thing
Vowels			
High Front Vowel with duration	iy	i_long	keep
Lax Front Vowel	ih	ic	him
High Back Vowel with duration	uw	u_long	blue
Lax Back Vowel	uh	hs	push
Mid-low Front Vowel	eh	ep	nest
Mid-low Front Vowel with duration	*eh	ep_long	fairy
Central Vowel with duration	er	epr_long	turn
Rounded Mid-low Back Vowel	*ao	ct	*Hartenbos*
Rounded Mid-low Back Vowel with duration	ao	ct_long	bore
Low Back Vowel	oh	ab	hot
Lax Mid-low Vowel	ah	vt	hut
Low Back Vowel with duration	aa	as_long	harp
Central Vowel (Schwa)	ax	sw	the
Mid-low Front Vowel	ae	ae	average
Mid-low Front Vowel with duration	*ae	ae_long	dad
Diphthongs			
	aw	a_lnkhs	about
	oy	ct_lnkic	appoint
	ey	e_lnkic	able
	ea	ep_lnksw	affair
	ua	hs_lnksw	cruel
	ia	ic_lnksw	luckiest
	ow	sw_lnkhs	show
	ay	vt_lnkic	fright

Table B.1: *The full ARPABET phoneme set used.*

Appendix C

Allowed grapheme-phoneme matches

The correspondences used when using hand-aligned matches to align the dictionary are shown in table C.1. Double phonemes are joined with a + symbol.

a	aa	ae	ah	ao	aw	ax	ax+ih	ay	ea
	eh	eh+ax	eh+iy	er	ey	ey+y	ih	iy	oh
	ow	oy	ua	uh	w+aa	w+ae	w+ah	w+eh	w+ey
	w+ax	y+aa	y+ae	y+ah	y+ax	y+ey	y+ih	sil	
b	b								
c	ch	g	jh	k	s	sh	t+s	z	zh
d	d	jh	t						
e	aa	ae	ah	ax	ea	eh	eh+ax	er	ey
	ia	ih	iy	oh	ow	ua	uh	uw	w+ax
	w+ay	w+eh	w+er	w+ih	w+iy	y	y+ax	y+eh	y+ih
	y+iy	y+uh	y+uw	sil+eh					
f	f	v							
g	ch	f	g	jh	k	sil	zh		
h	ax	f	hh	th	ow				
i	ae	ah	ax	ay	ay+ax	ia	ih	ih+ax	eh
	er	iy	w+ah	w+ax	w+ay	w+ih	y	y+ax	y+ih
	y+iy								
j	d	jh	hh	y	zh				
k	k								
l	ah+l	ax+l	l	uh+l	y				
m	ah+m	ax+m	ih+m	m					

n	ah+n	en	m	n	n+y	ng			
o	aa	ae	ah	ao	aw	ax	ax+uw	er	hh+w
	ih	oh	ow	ow+ax	oy	ua	uh	uw	w
	w+aa	w+ah	w+ax	w+ay	w+ao	y+aa	y+ax	y+ow	sil+aa
p	f	p	v						
q	k	k+w							
r	ax+r	er	er+r	r	r+ah	r+ax	r+ow	y	
s	ah+z	ch	ih+z	s	sh	z	zh		
t	ch	d	dh	jh	s	sh	t	th	zh
u	aa	ae	ah	ao	aw	ax	ay	eh	er
	ia	ih	iy+ax	ua	uh	uw	w	w+ax	y
	y+ah	y+ao	y+ax	y+er	y+ua	y+uh	y+uw		
v	f	v							
w	hh+w	sil	v	w	wh				
x	g+z	g+zh	k+s	k+sh	k+z	sh	z		
y	ah	ax	ay	ih	iy	y			
z	s	t+s	t+z	z	zh				
’	ax	ih							

Table C.1: *Hand-aligned grapheme-phoneme mappings.*

Appendix D

G2P phoneme errors

Table D.1 gives a summary of the confusion matrix of phoneme prediction errors during G2P conversion. The total number of times a given phoneme occurred in the test data is given, as well as the phonemes it was most often incorrectly predicted as (with the number of false predictions for each of those phonemes). Only phonemes with an accuracy of less than 90% are listed.

Phoneme	Correct (%)	Total occurences	Erroneously predicted as
uh	61.2	884	uw(101), ax(76), Del(62)
iy	71.9	2012	ih(195), eh(125), ax(122), ay(38)
y	72.9	1308	Del(103), ch(43), ih(36)
zh	75.4	134	jh(17), z(7), sh(6)
dh	73.3	157	th(39)
ow	77.1	1703	ax(144), aa(123), Del(27), aw(24), uw(23)
aa	78.9	3665	ax(331), ow(130), ae(99), ah(40), ao(33), ey(33)
ay	80.4	1978	ih(223), ax(42), iy(31)
ax	80.5	16095	ih(837), aa(374), eh(319), Del(302), ae(302)
eh	80.7	4362	ax(332), iy(149), ih(147), Del(77)
ae	81.2	3174	ax(332), ey(110), aa(82)
uw	81.6	1379	uh(101), ax(37), ow(30), ah(29), Del(23)
aw	82.5	543	ow(43), Del(9), ax(7), uw(6)
er	82.6	984	ax(130)
ch	93.0	652	sh(30), y(27), k(25)
ey	84.2	2792	ae(133), ax(123), aa(44), ih(41), iy(29)
ah	84.5	2100	ax(117), aa(49), uh(43), uw(28), ow(21)
ih	85.6	12998	ax(872), ay(244), eh(187), iy(165), Del(146)
z	86.5	4017	s(500)
ao	87.5	1331	ax(63), aa(40), ae(20)
w	89.8	1311	Del(31), uw(13), uh(9)
Insertions		1047	ax(217), y(126), ih(122), t(71), eh(58), w(47)

Table D.1: *Phoneme errors during G2P conversion.*

Appendix E

Optimal GP2P training words

The following list gives the first 60 words of the training set determined to give optimal GP2P conversion accuracy.

1. unsympathetic
2. materialized
3. refurbishing
4. captivated
5. physiology
6. Yugoslavia
7. noteworthy
8. harpsichordists
9. searcher
10. formulation
11. outsider's
12. enjoying
13. ceremonies
14. edwardian
15. nettle
16. intrusion
17. upsides
18. wand
19. overladen
20. fairly
21. authoress
22. Yugoslav
23. lifeline
24. effectiveness
25. Joshua
26. body's
27. ravine
28. yeah
29. warrior
30. versifiers
31. aerial
32. officialdom
33. terminal
34. flotsam
35. curate
36. colling
37. shocks
38. mildness
39. full
40. scrutinized
41. cuisine
42. eradicate
43. respecting
44. instinctively
45. Bradbury
46. carefully
47. guardsman's
48. physique
49. handsworth
50. educationally
51. fluctuates
52. fortune
53. milked
54. midday
55. Edmunds
56. Cossacks
57. lowbrows
58. bedclothes
59. oaths
60. fawn