UNIVERSITY OF SOUTHAMPTON

Extending Pronunciation by Analogy for Speech Synthesis Applications

by

Tasanawan Soonklang

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

February 2008

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Tasanawan Soonklang

Automatic pronunciation of unknown words, especially in English, is a hard problem of great importance in speech technology. This thesis focuses on a data-driven approach namely 'pronunciation by analogy', so-called PbA, for generating the pronunciation of unknown words from input text. The aim is to explore many useful aspects of the use of PbA in speech synthesis applications. This thesis is mostly devoted to the problem of proper name pronunciation, because previous work showed that proper names have significant impact on the performance of text-to-speech (TTS) systems. The extension of PbA for multilingual pronunciation is also studied.

The performance of PbA is investigated in a wide variety of aspects including: to incorporate automatic syllabification by analogy, to determine the effect of different kinds of lexicon, to determine the effect of lexicon size, to test with seven European languages in order to quantify the relationship between transcription accuracy and orthography, and to compare with other data-driven methods in terms of objective and subjective evaluations.

The experimental results show that PbA can achieve a promising level of word accuracy and is superior to other methods tested on the problem of proper name pronunciation. In the objective evaluation, the best performance is 68.38% names correct and 94.31% phonemes correct, with a standard PbA using a leave-one-out strategy on 52,911 names in the CMU dictionary. In the subjective evaluation, the comparison is primarily based on 24 listeners' opinions of the acceptability of pronunciations from 150 names. Wilcoxson signed-rank tests show that the dictionary pronunciations are rated superior to the automatically-inferred pronunciations; one part of listening tests shows that PbA is marginally superior to the other methods, but no such superiority is seen for another part of listening tests. With reference to the performance on seven European languages (Dutch, English, French, Frisian, German, Norwegian, and Spanish), PbA achieves more than 85% words correct in case of all languages except English. In conclusion, this thesis has shown that PbA should become the method of choice in TTS applications.

Contents

List of Figures

List of Tables

Acknowledgements

During the years of this research, I have had cause to be grateful for the advice, support and understanding of many people. In particular I would like to express my sincere appreciation and gratitude to my supervisor, Prof. Bob Damper, for his continuous support throughout this research. In the early stage of the research, I have benefited greatly from the advice and comments of Dr. John Carter. Many thanks to Marchand Yannick for his SbA source code and the advice about programming techniques. John-David S Marsters for the alignment of all lexicons. Vincent Pagel for his CART source code. Alan Black for a list of proper names in CMUDICT. Richard Sunderland for making an early start on the Festival system. Jenni and Sarah for reading some of the early drafts and patiently took time to edit them. As well as the people mentioned above, a mention must also go to many friends in ISIS research group and Thai Society in Southampton for their encouragement. I am deeply indebted to my family and my boyfriend for all their mental support. Lastly, this thesis cannot be a success without a great support from the Royal Thai Government for full funding during my PhD study.

To my beloved parents.

Chapter 1

Introduction

Text-to-speech (TTS) synthesis is a computerised system for converting printed text into synthetic speech. TTS synthesis, which has developed enormously over the past decade, is an emerging technology with many important applications in next-generation information systems (Klatt 1987; Dutoit 1997, p.30). These applications have gradually become an important feature in our daily lives. A good example for the use of TTS today is directory assistance in telecommunication services, which can convert textual information into voice and respond to a request from a customer over the telephone. TTS systems can be used to read aloud the text for the blind and dyslexic. Furthermore, they have been used for reading e-mails, news, travel directions, and other information in a wide range of applications. They have also been used in computer-aided learning systems for students who learn a new language.

A typical TTS system can be divided into two major modules: a natural language processing (NLP) module, and a digital signal processing (DSP) module (Dutoit 1997, p.14; Ng 1998, p.3). The NLP module takes input in the form of text and outputs a phonetic transcription together with the prosody as the symbolic linguistic representation. Next, the DSP module takes the symbolic linguistic representation as input and outputs the synthesised speech waveform. The diagram of a simple TTS system is shown in Figure 1.1.

The NLP module first separates the text steam into clauses or sentences. Then, it

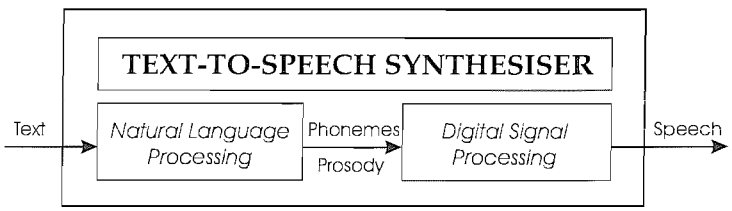

FIGURE 1.1: A simple functional diagram of a TTS system. (Redrawn from Dutoit 1997, p.14).

performs text normalisation, where numbers and abbreviations from the raw text are converted into their full word equivalents. After normalisation, the next process called text-to-phoneme conversion or grapheme-to-phoneme conversion, where phonetic transcriptions are assigned to each word, is performed. Next, the NLP module performs syllabification, and assigns the prosodic information, mainly comprised of intonation and stress. The combination of phonetic transcriptions and prosody forms the symbolic linguistic representation, and is an input into the next module. The second module, often referred to as a synthesiser, takes the symbolic linguistic representation and converts it into speech sounds. This thesis focuses on the text-to-phoneme conversion in the NLP module.

Text-to-phoneme conversion is a procedure for mapping a spelling of a word into a string of phonetic symbols that represents a pronunciation of the word. This process is an integral part of TTS synthesis, and is also an important part of speech recognition. The most obvious and effective approach is simply to look up pronunciations of input words—or, perhaps, morphemes after morphological decomposition—in a dictionary. This will work very well, but only provided the word is actually in the dictionary. However, it is impractical (strictly, impossible) to store all the words of the language, since this constitutes an open set. Thus, the dictionary approach can not be a complete or sufficient solution to this problem; some 'back-up' procedure is needed for words not in the dictionary. The usual approach is to employ a set of phonological (letter-to-sound, or text-to-phoneme) rules written by a linguist or phonetician, with expert knowledge of the target language, as a back-up or secondary strategy to the primary strategy of dictionary look-up. Since the early 1980's, the problem of automatic pronunciation has been considered as a pattern matching problem in the machine learning community, and a data-driven approach has emerged as the next generation of back-up strategy. Several current researchers attempt to create a new data-driven method or apply an existing approach to convert letters into phonemes.

1.1 Problem Statement

One of the biggest challenges for text-to-phoneme conversion is the automatic pronunciation of unknown words, i.e., the words which are not listed in a dictionary. Unknown words may be common words, typos, neologisms or proper names. An obvious way to address the problem is to compose manually a dictionary with acceptable pronunciations which is then consulted during synthesis. In general, however, this solution can never be complete as the possibility always exists of encountering unknown words in the system input that are not listed in the dictionary. Hence, a secondary or back-up strategy to the primary strategy of dictionary look-up must be devised. Many rule-based methods have been proposed as initial attempts to predict word pronunciation from the spelling. To create letter-to-sound (LTS) rules, expert linguists are needed to write the

rules carefully. The hand-crafted rule-based approach has often been the method of choice. Nevertheless, this knowledge-based approach is highly language-specific (Sproat 1997, p.83), and has to be redone at some expense for each new language. Moreover, there is good evidence that manually-written LTS rules work very poorly compared to data-driven methods, certainly for English (Damper et al. 1999). Therefore, a data-driven technique is considered attractive for modern TTS systems, i.e., learning automatically from data. Numerous data-driven techniques have been proposed to deal with the problem of automatic pronunciation. In this thesis, an existing data-driven approach named pronunciation by analogy (PbA) is studied to deal with this problem in speech synthesis applications. PbA is selected for this study since, for some years now, it is well-documented to have easily the best performance relative to a variety of competitors (Damper et al. 1999).

This thesis focuses on pronunciation of unknown words, particularly proper names, for several reasons. First, many commercial applications often require the pronunciation of proper names, i.e., the names of people, streets, cities, places, companies, etc. Second, prior work on large word lists in this area showed that proper names have significant impact on the performance of the speech synthesis system (Vitale 1991; Font Llitjós and Black 2001). Finally, the problem of proper names is a special challenge because the geographical and language origin of the names can be varied, and the number of distinct names is very large (Vitale 1991; Spiegel 2003). A few studies have proposed various techniques to solve the problem of proper names, but they have not achieved a high level of accuracy (Fackrell and Skut 2004). A proper name in this work means that the name is written in English, but the origin of the name may be from languages other than English.

Syllabification is also an important process in the NLP module. As observed by Marchand and Damper (2007), integrating syllable boundary information manually in the orthographic input can dramatically improve the performance of pronunciation by analogy for common words. However, such information is not available in a dictionary of proper names. To investigate this phenomenon in proper names, automatic syllabification has been evaluated using the models of syllabification and pronunciation by analogy.

As mentioned above, unknown words typically include proper names and common words that have not been listed in the lexicon. In practice, when given an unknown word, we would not know if it is a proper name or a common word. Automatic classification of unknown words is possible to be developed, however, the potential errors must be taken into account. Therefore, the performance of PbA is investigated when the system infers a pronunciation by analogy with different dictionary compositions: (i) common words only, (ii) proper names only, and (iii) a mixture of the two. If PbA can achieved a high accuracy in case of a mixture, then there would be little or no advantage to attempting automatic inference of input-word class.

Most studies evaluate the performance objectively, which means that the pronunciation generated by their models is taken as correct if it is the same as that in a dictionary that is taken as a 'gold standard'. However, a crucial limitation for any TTS system is the problem of automatically generating 'acceptable pronunciations'. In terms of acceptable pronunciations, it means that the pronunciation should be acceptable to potential users of a TTS system. Different people will have different opinions about the extent to which a proper name should conform to the pronunciation conventions of the local speech community, and the preferred pronunciation may vary between name owners/origins. Thus, subjective evaluation is also important, especially for proper names. Subjective evaluation is difficult to conduct because there is no obvious criterion of correctness, and cannot deal with large numbers of words. Therefore, only few studies have conducted listening experiments. In this thesis, the performance of our chosen method is compared to that of the other data-driven models for automatic pronunciation in both subjective and objective ways.

In terms of multilingual synthesis, the difficulty of the pronunciation problem depends on the complexity of spelling-to-sound mappings according to the particular writing system of the language. Hence, the degree of success achieved varies widely across languages and also across dictionaries, even for the same language with the same method. Further, the sizes of the training and test sets are an important consideration in data-driven approaches. In this thesis, the variation of letter-to-phoneme transcription accuracy across seven European languages has been studied, mostly for common words. Also, the relationship between the size of dictionary and the accuracy obtained has been investigated.

1.2 Contributions

The main contribution of this research is to extend PbA for speech synthesis applications. The term 'extend' is used for two reasons. Firstly, this thesis contains some original work using PbA, but is not just simply repetition or summary of the existing method. Secondly, it aims to explore many useful aspects of the use of PbA for TTS applications. The primary contribution of this work has been mainly involved with the pronunciation of proper names. Specifically, the important contributions are:

- Literature relating to text-to-phoneme conversion and syllabification is reviewed. The use of text-to-phoneme conversion in automatic pronunciation of proper names is highlighted.
- Four different data-driven methods are evaluated objectively with a list of American proper names. Pronunciations derived from a dictionary itself and from three data driven methods: PbA, decision tree, and table look-up, are selected for subjective evaluation. These methods are compared and the results are discussed.

- Using the idea of syllabification by analogy for syllabifying and inferring pronunciation for proper names, experiments are conducted to investigate whether or not the performance of PbA improves when using the model of syllabification together with pronunciation by analogy.
- Extending PbA for multilingual pronunciation, 7 European languages are evaluated using 12 different lexicons. Ways to quantify the variation of transcription difficulty across the deep/shallow continuum of orthography are highlighted. The relationship between lexicon size and the accuracy obtained is also studied.
- Different lexicon compositions have been tested with PbA. These are discussed with reference to the problem of automatic classification of unknown words (common words or proper names). The performances of PbA are investigated, when inferring from a dictionary of the same/different class of input word and also from a mixture of the two. The results are compared and discussed with regard to the possibility of avoiding automatic word categorisation.

The work in this thesis has contributed in part or full to the following publications:

- Soonklang, T., Damper, R. I. and Marchand, Y. (2007). Multilingual pronunciation by analogy. *Natural Language Engineering*. Submitted.
- Damper, R. I. and Soonklang, T. (2007). Subjective evaluation of techniques for proper name pronunciation. *IEEE Transactions on Audio, Speech and Language Processing*. In press.
- Soonklang, T., Damper, R. I. and Marchand, Y. (2007). Effect of lexicon composition in pronunciation by analogy (Speech). In *Proceedings of 10th International Conference on Text, Speech and Dialogue (TSD 2007)*, pp. 464–471, Pilsen, Czech Republic.
- Soonklang, T., Damper, R. I. and Marchand, Y. (2005) Comparative objective and subjective evaluation of three data-driven techniques for proper name pronunciation (Poster). In *Proceedings of Interspeech 2005*, pp. 1905–1908, Lisbon, Portugal.
- Damper, R. I., Marchand, Y., Adsett, C. R., Soonklang, T. and Marsters, J. D. S. (2005) Multilingual data-driven pronunciation (Speech). In *Proceedings of 10th International Conference on Speech and Computer (SPECOM 2005)*, pp. 167–170, Patras, Greece.

1.3 Structure of the Thesis

This prelude has introduced the basics of a TTS system and its components. The problem of text-to-phoneme conversion has been stated, and has highlighted the difficulties of unknown word pronunciation, particularly for proper names written in English text.

A literature review on text-to-phoneme conversion is described in detail in Chapter 2. Various techniques are reviewed and discussed. Previous studies involving automatic pronunciation of proper names have been addressed.

Chapter 3 introduces the principles of pronunciation by analogy. PbA is one of the most successful backup strategies that exploit the phonological knowledge implicit in the dictionary of known words to generate a pronunciation for an unknown word. A review of many variants of PbA with their results on different datasets mostly for common words is provided. This chapter illustrates the process of PbA with a step-by-step example.

Chapter 4 introduces the important topic of syllabification. A review of previous work dealing with this problem is presented. Syllabification algorithms using the analogy concept are described. The results of applying the series model for automatic syllabification and pronunciation by analogy, in which syllabification is followed sequentially by pronunciation generation, are presented and discussed.

A further potential advantage of data-driven approaches is that they are highly portable between different languages, only provided a database (or lexicon) of words and their pronunciations is available. In Chapter 5, seven European languages have been evaluated using a PbA method. The results are presented and discussed. Much of this discussion suggests the idea that the transcription accuracy maybe relates to the depth of orthography of a language. The effect of lexicon size on accuracy is also considered in this chapter. Conducting experiments on different sizes of lexicon would give an idea of the reasonable size of lexicon that should be used to compromise a trade-off between the performance and the processing time of the PbA approach.

It remains to be decided whether or not the unknown word needs to be classified in advance as a common word or proper name. Chapter 6 is concerned with investigating the effect of lexicon composition: common words only, proper names only, and a mixture. Experiments are conducted to see how performance of PbA varies for different kinds of words when inferring from different lexical databases. These are illustrative of the problems that are encountered when miss-classification happens, or no prior classification is provided. If good results can be obtained on the mixture dictionary of common words and proper names, comparable to those on common words or proper names alone, these would suggests that there may be no need for automatic word categorisation (common word vs. proper name) to be attempted, with its attendant dangers of mis-classification.

Chapter 7 and Chapter 8 are devoted to the comparison of PbA with other data-driven

methods. Objective and subjective evaluations are provided with discussion. The main focus of interest in this chapter is the problem of proper names written in English. The quality of a pronunciation model is judged by comparing with the standard dictionary and by assessing the pronunciations by humans. Re-implementations of each method are described in Chapter 7. The listening tests are conducted and described in Chapter 8. The pronunciation of the dictionary itself is also included and compared with those of data-driven methods in the listening tests because it is generally used as the primary strategy in any practical TTS system. The results are presented and discussed in a statistically meaningful sense.

The thesis concludes with Chapter 9, which highlights the overall contributions, and provides some ideas for future work.

Chapter 2

Text-to-Phoneme Conversion

2.1 Introduction

In a TTS system, after normalising the text, phonetic transcriptions are assigned to each word by the text-to-phoneme conversion module (also known as 'grapheme-to-phoneme', or 'letter-to-sound', these terms will be used interchangeably in this thesis). The module maps the spelling of a word into a string of phonetic symbols. Phonetic transcription can be viewed as a symbolisation, mapping sounds into discrete context-free symbols. It represents the pronunciation of a word, and therefore it is an essential part of speech synthesis systems and automatic speech recognition (ASR). For example, phonetic transcription can then be combined together with prosody and feeds as an input to digital signal processing in a TTS system to create the speech sound. In an ASR system, phonetic transcription can be provided as a reference transcription for the words in the vocabulary.

There are several strategies to determine the pronunciation of a word from its spelling. These are commonly classified into three broad categories: dictionary look-up, hand-crafted rule-based, and data-driven approach. These strategies, together with their previous researches in speech synthesis applications will be reviewed in the following sections. However, the data-driven approach is the main focus here due to the promising preliminary results of the pilot research study, which is not reported here.

2.2 Dictionary Look-Up Approach

In the dictionary look-up approach, a dictionary is generated and used to map the orthographic form of a word into the phonetic form. To make it applicable on a large scale, modern TTS systems exploit a dictionary containing root and affix forms of words called morphemes (Dutoit 1997, p.111; Holmes and Holmes 2001, p.100). In this way, the

dictionary can be kept to a reasonable size. The concept of a morpheme dictionary was first proposed by Lee (1969). A morpheme is the smallest unit of meaning that a word can be divided into. For example, the word *unlikely* contains three morphemes: *un*, *like*, and *ly*. The pronunciation of an input word is determined by looking up the input word in the dictionary. If it is found, the phonemes with the pronunciation specified in the dictionary are provided. If it is not, morphemic decomposition is performed and the morphemes are searched in a dictionary. Then morphological rules are used to combine the pronunciation of the input word with the pronunciation of its component morphemes. Allen et al. (1979) followed this concept in developing an algorithm for morphemic decomposition in MITalk, containing tens of thousands of morphemes. The AT&T Bell Laboratories TTS system also operates using this principle (Dutoit 1997, p.111).

This approach is the simplest and the most reliable method; it has the advantage of being quick and accurate (Vitale 1991). However, it completely fails when the pronunciation cannot be determined by using a dictionary. The set of all words of a language is unbounded due to the existence of neologisms, proper names, and compounds. Thus, it is impossible to list all words in a language in a dictionary. The occurrence of out-of-vocabulary (OOV) words is inevitable in this case. Actual investigations reported OOV rates within a 20k newspaper corpus in five languages: 2.5% for English, about 4% for Japanese and Italian, 5.8% for French and 10% for German (Matsuoka et al. 1996). This occurrence within a speech synthesis application is very harmful for the user's acceptance (Muller et al. 1996). Hence, a 'backup' strategy, such as LTS rules and data-driven approaches, is required to guess a pronunciation of an unknown word, while the dictionary look-up method is usually used as a primary strategy to determine the pronunciation of a known word.

2.3 Hand-Crafted Rule-Based Approach

In early TTS systems, the derivation of word pronunciation was principally focused on the rule-based approach (Sproat 1997, p.83; Holmes and Holmes 2001, p.96). Rules of pronunciation are applied to input words to find out their pronunciations based on their spellings. This approach is well known as LTS rules. With this strategy, most of the phonological knowledge of dictionaries is transferred into a set of LTS rules. This set of hand-crafted rules requires extensive knowledge from expert linguists. Each rule specifies a phoneme corresponding to one or more letters. The set of rules is used to guess the pronunciation of any word by giving a letter string's context to determine which rule should be applied. The rules are usually comprised of four parts in the form:

$$A[B]C \rightarrow D.$$

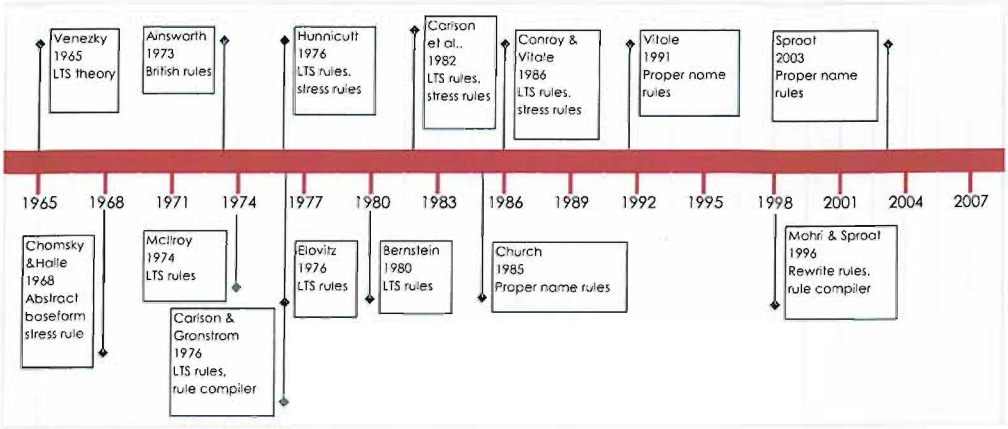

FIGURE 2.1: Historical timeline of linguistic rule-based algorithms in TTS systems.

The meaning is that character string B, occurring with left context A and right context C, gets a pronunciation defined by the phonemes D. The most specific rule is typically at the top, and the most general rule at the bottom of an ordered list. The pronunciation of a word can be found if the rules are applied to each letter in the string normally from left to right; however, some rules may operate from right to left. The rules are searched for each a target string, starting with the first letter of the word. If the matching text is found, and the right and left context patterns also match, the corresponding phonemes D for that rule are output and the next untranscribed letter is taken as the target.

Several studies have proposed rule sets for TTS systems over the past decades. The historical timeline of selected rule-based approaches is presented in Figure 2.1. An excellent review of these techniques prior to 1987 can be found in the work of Klatt (1987). More recently, Spiegel (2003) reports spending up to 15 years improving the pronunciation of proper names using rules. The rule-based approach persisted because of a belief that it can outperform the other techniques in generating pronunciations, despite being costly to develop and very time-consuming (Sproat 1997, p.75). An example of a publically-accessible set of rules is that of Elovitz et al. (1976) which is expected to achieve approximately 90% words correct in a random sample of English text. Bagshaw (1998) and Damper et al. (1999) evaluated this claim by using different datasets; however, they did not achieve such a high accuracy, the first finding around 20% words correct with the latter finding 25% words correct. The main reason for this considerable discrepancy is probably the difference between their scoring methods. Elovitz et al. (1976) used frequency weighting and based their scoring on the listeners' acceptance of "good" pronunciations, while the others based their scoring strictly on 'correct' pronunciations, identical to the dictionary entries. One of the drawbacks of the rule-based approach is that the rules need to be created from scratch when a new language is included (Daelemans and van den Bosch 1997). Additionally, although the rule-based approach can work on any input, the complexity of the rules grows substantially when irregular spellings or pronunciations are taken into account. This

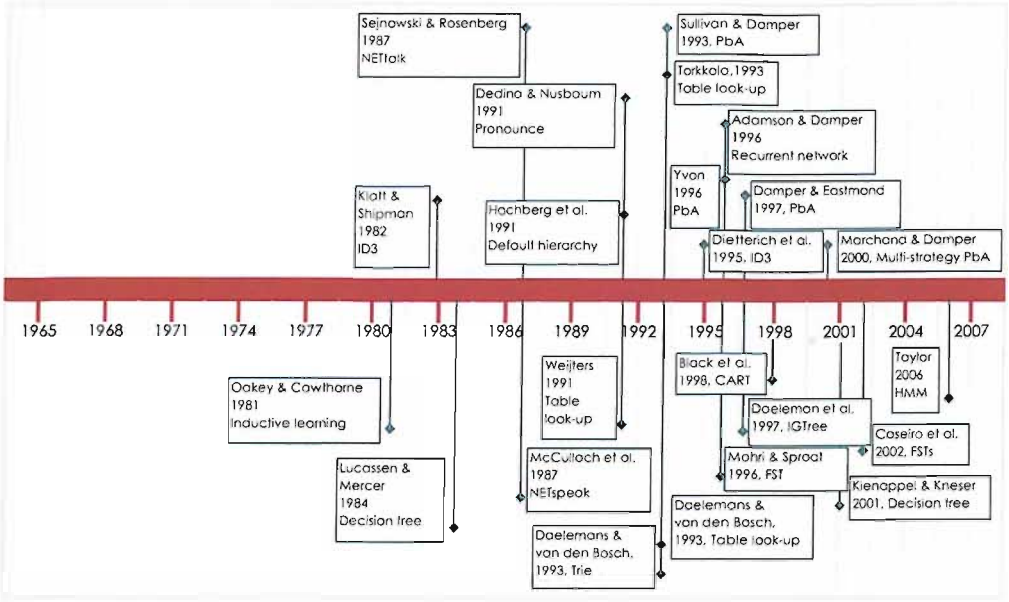

FIGURE 2.2: Historical timeline of the data-driven approach in TTS systems.

is especially true for the English language, which has a complex spelling-to-phonology correspondence. Both dictionary look-up and rule-based approaches have advantages and disadvantages. Therefore, many TTS systems use a combination of both approaches in text-to-phoneme conversion (Vitale 1991), e.g., in MITalk (Allen et al. 1987, p.12-13) and DECtalk (Hallahan 1996). Another solution is to use a combination of dictionary look-up and data-driven approaches, which is discussed in the next section.

2.4 Data-Driven Approach

In most modern TTS systems, the dictionary look-up approach is used as the primary strategy, and the rule-based method as the backup strategy. Recently, the data-driven approach has been focused upon as a promising backup strategy for transcribing words that are not in the dictionary (Marchand and Damper 2000). The data-driven approach for text-to-phoneme conversion is a machine learning technique that implicitly extracts knowledge from training data, and then exploits it to convert text to phonemes automatically. Since 1987, the emergence of NETtalk by Sejnowski and Rosenberg (1987) has provided an inspiration in applying data-driven methods to speech synthesis, although NETtalk was not the first attempt to apply machine learning to text-to-phoneme conversion (the first attempt was a rule induction – see later in Section 2.4.4). The availability of machine-readable pronunciation dictionaries and the development of the machine learning model have also catalysed the evolution of this system (Damper 2001, p.1). Furthermore, this is a key element for the design of multilingual TTS systems, since the data-driven approach is essentially language-independent.

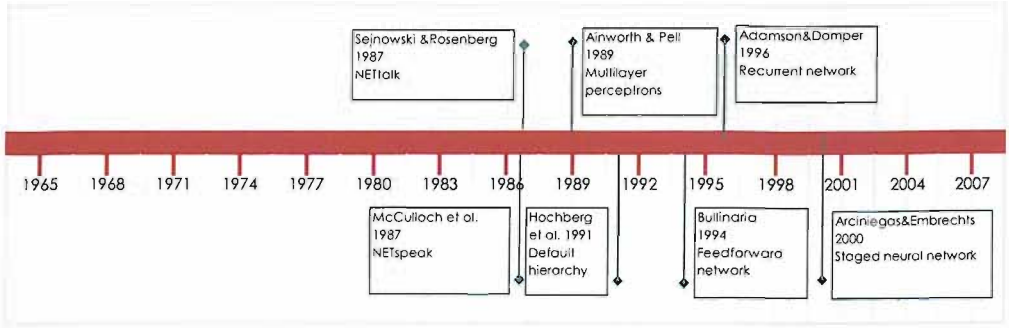

FIGURE 2.3: Historical timeline of the neural network method in text-to-phoneme conversion.

During the past decade, a large number of data-driven methods have been proposed for text-to-phoneme conversion, such as decision trees, neural networks, and analogy-based techniques. The development of these systems has been surveyed and is partially presented chronologically in Figure 2.2. An excellent survey of these techniques prior to 1994 can be found in the reviews of Klatt (1987) and Yvon (1994). Data-driven techniques in the diagram are categorised and reviewed in this section. All techniques reviewed here are *supervised learning* methods in which the phoneme (output) to be predicted is provided during a learning stage. From this point forward, the training data will refer to pairs of orthographic and phonemic form.

2.4.1 Neural Networks

Neural networks, so-called connectionist architectures, have been used to learn the correspondence between phonemes and the letters of each word. In the learning process, training data are encoded into a set of connection weights and thresholds between the nodes. Examples of this approach are NETtalk (Sejnowski and Rosenberg 1987) and NETspeak (McCulloch et al. 1987), which were trained by backpropagation to convert text to phonemes.

NETtalk requires text pre-aligned with its corresponding phonemes. Each word in a dictionary was converted into a sequence of 7-letter window. Binary coding of 29 bits was used to represent a 7-letter context, 26 bits for English alphabets, plus three additional bits for punctuation and word boundaries. Thus, the number of input units was 203 (7×29). The number of output units was 26, representing the phonemes in terms of 21 articulatory features, such as the location of articulation point, voicing, vowel height, plus five additional units for stress and syllable boundaries. Each word in the dictionary was stepped though the window letter-by-letter, and encoded into input and output units. Then these units were fed into the networks to compute an output, and the weights were adjusted after each word so as to reduce the error. The output was given as a vector of 26 output activations (between 0 and 1). The 21-bit phoneme

code that made the smallest angle with the output activation vector was chosen as the output. The network was evaluated with the 20,012 words of Webster's Pocket Dictionary. The performance was reported to be about 90% phonemes correct by using 120 hidden units trained on the 1,000 most commonly occurring words, and tested with the whole dictionary, after five training passes. Various numbers of hidden units and layers were studied, and it was found that the performance improved when increasing hidden units and layers. Different sizes of window, ranging from 3 to 11 letters, were also studied, and it was found that the performance improved with the size of the windows. NETtalk is a very well-known data-driven approach, since then the problem of English text-to-phoneme conversion has become a benchmark for machine learning (Damper 2001, p.11). Neural networks have been extensively studied and various results have been presented. A selected survey of neural networks in text-to-phoneme conversion is shown chronologically in Figure 2.3.

A re-implementation of NETtalk was studied by McCulloch et al. (1987), exploring the impact of different input and output encodings using a different dictionary and output phoneme set. In 1989, this architecture has been tested again with a large data set of 70,000 words (Ainsworth and Pell 1989). However, the performance in terms of phonemes correct hardly exceeds 90%.

This approach is not limited to multilayer perceptrons. Attempts have been made to use alternative network designs. For example, Hochberg et al. (1991) presented an approach they called 'default hierarchy', using rules like an expert-made system, but trained automatically like a neural network. In training, specific rules were learned only if they were exceptions to general rules. Using the hierarchy, default rules were used when no relevant specific rules were found. After training on 18,008 words, the model achieved 90% phonemes correct on an unseen 2,000-word test set. Another example is a self-organising neural network, which was developed for a multilingual pronunciation dictionary project (Hensen 1994).

Various neural networks have also been adapted to cope with the transcription of non-aligned data such as a syntactic neural network (Lucas and Damper 1992), the extension of NETtalk (Bullinaria 1994), and recurrent neural networks (Adamson and Damper 1996). More recently, Arciniegas and Embrechts (2000) used a series of two-staged neural networks for converting text to phonemes. The first stage learns which one or two phonemes were represented for each letter, so that different networks can be used at the next stage, to learn the letter to phoneme mapping. They also introduced a new window positioning structure, in which there was an unequal number of letters before and after the target. The algorithm was trained on the 2,000 most common words in American English. It achieved a letter accuracy of 97% on an unfair test because the training and test set are the same.

In summary, neural network classifiers work reasonably effectively for the classification

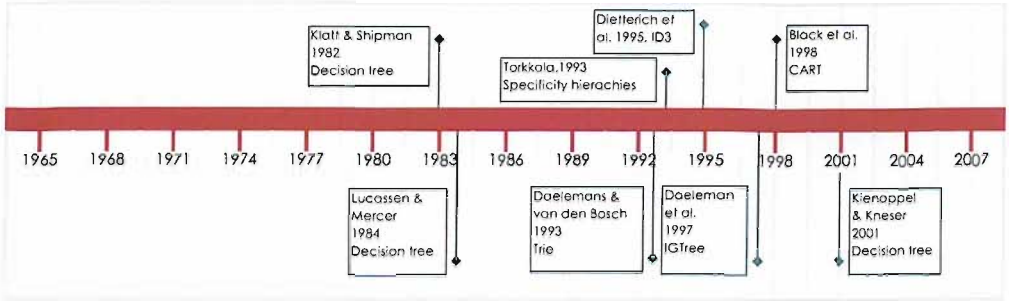

FIGURE 2.4: Historical timeline of the decision tree in text-to-phoneme conversion.

problem. Thus, there are several attempts for predicting phonemes by using neural networks. Neural networks have been used to model the transformation between letter sequence and phonetic sequence by training on a pronunciation dictionary. Various results reported here tend to demonstrate that the performance of such attempts is limited. Despite the high accuracy in terms of phonemes correct often reported, some studies showed that the performance of this architecture has not yet achieved a high accuracy in terms of words correct, and is considered inferior to that of the other data-driven approaches (Golding and Rosenbloom 1993; Dietterich et al. 1995; Damper et al. 1999). One drawback of applying neural networks to solve these problems is that the trained networks are often viewed as a black box, because of a lack of explanation of the inner workings and a lack of the capability to extract knowledge to gain better understanding of the problems.

2.4.2 Decision Trees

A clear disadvantage of the linguistic rule-based strategy is that it requires expert linguists to build a set of hand-crafted rules. Thus, the trained rule-based technique has emerged to build LTS rules automatically. One of the most popular techniques is a decision tree learning method such as ID3 (Quinlan 1987), in which the learned function is represented by a decision tree. ID3 builds a classification tree from a given set of classified instances, then this decision tree can be used to classify an unknown instance. To build a decision tree, starting at the root node of the tree, an instance is classified into two groups (child nodes) by testing or questioning the attribute specified by this node, then this process is repeated iteratively for the subtree rooted at the child node until each leaf node corresponds to the class of the instance. Decision trees can also be represented as sets of if-then rules. In the case of text-to-phoneme conversion, decision trees can be used to predict phonemes from examples of aligned training data. The use of decision trees has been reported by many researchers as can be seen in the chronological diagram in Figure 2.4.

In 1982, Klatt and Shipman first proposed the concept of decision tree to solve the problem of text-to-phoneme conversion. With their model, induction rules were created

from each instance of each letter in a 20,000-word English dictionary by using left and right context of the letter. Then decision trees were built in the form of an ordered list of context-dependent rules for rapid execution. This work can also fall into the category of rule induction. A fixed-size input window, a learning pattern similar to NETtalk, has been used to build a search tree recognition model from a 50,000-word dictionary (Lucassen and Mercer 1984). The results were reported in terms of phonemes correct from 5,000 random-selected words, and achieved an accuracy of 94% which was superior to that of NETtalk. Later in the work of Daelemans and van den Bosch (1993), a trie (a special kind of tree structure) was built from the training data. The first level of the tree corresponds to the focus letter and the other levels correspond to letter context. This can be viewed as a hierarchical description of rules.

Other attempts of using ID3-like techniques with multilingual lexicons were successfully reported with promising results (Torkkola 1993; van den Bosch and Daelemans 1993; Black et al. 1998). For example, Black et al. (1998) proposed to use classification and regression tree (CART). In their work, for each letter in the alphabet of the language, CART was trained by giving the focus letter, together with three context letters on either side, to predict the phoneme. In their best case, the automatic LTS model achieved a comparable or higher accuracy than the manual LTS rules. CART has been applied to four lexicons, including Oxford Advanced Learners Dictionary (UK English), CMUDICT (US English), BRULEX (French), and Celex Lexicon (German), where it achieved 75%, 58%, 93% and 89% words correct respectively. However, it was not quite clear how many words they used as a train set and test set as they wrote "We split the data into train and test data by removing every tenth word from the lexicon". CART has been used as the phonetic transcription process in Festival, a public domain system intended for speech synthesis research available from `http://www.cstr.ed.ac.uk`. In the latest work of Kienappel and Kneser (2001), they proposed the decision tree technique for automatic grapheme-to-phoneme transcription in ASR systems. One UK-English name and two German lexicons (names and common words) were tested. The results were reported in terms of phoneme error rate and string error rate (SER). The best performance was 8% SER, when training on 320,000 words and testing on 16,000 words from the German common-word lexicon. The poorest performance was 33.7% SER, when training on 74,000 words and testing on 19,000 words from the English lexicon.

In summary, this approach uses decision trees built on a statistical basis from a number of context-to-phoneme pairs derived from a phonetic dictionary, to find the best feature sets of predicting phonemes with a certain probability. Comparisons of decision trees with other techniques in English have also been made (Dietterich et al. 1995; Damper et al. 1999; Damper and Soonklang 2007). The results showed that they performed substantially better than the rule-based approach and at about the same level as other data-driven methods such as table look-up and neural networks (approximately 60% words correct).

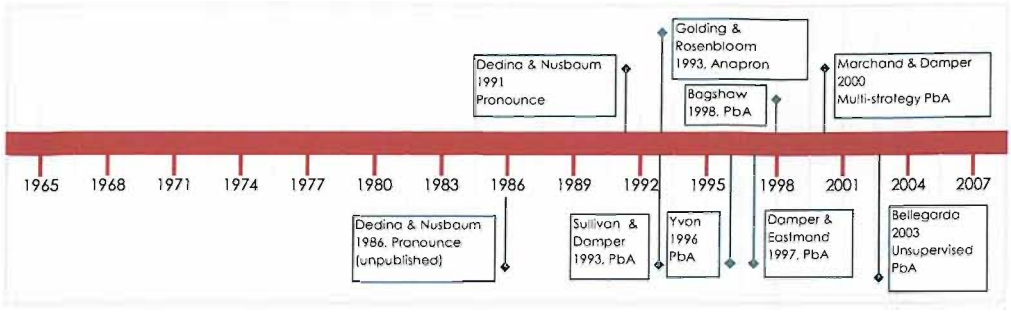

FIGURE 2.5: Historical timeline of the analogy-based approach in text-to-phoneme conversion.

2.4.3 Analogy Method

The analogy concept is basically developed by inference from observed human performance in unknown word pronunciation. In the process of reading aloud, an unknown word can be read on the basis of its analogy to known words whose spelling and pronunciation are familiar to the reader. This approach has emerged on the basis of the cognitive theories in the work of Glushko (1979), who introduced the model of reading by using an idea of a similarity matching component, that converts spelling strings of a given word to their pronunciation by matching them with lexical entries.

The most famous approach in this area is pronunciation by analogy (PbA). PbA uses the phonological knowledge from a dictionary by matching substrings of the input to substrings of lexical words, then collecting a partial pronunciation for each matched substring to create a directed graph. Later the phonemes along the shortest path are concatenated together to form the pronunciation. If there is only one candidate corresponding to a unique shortest path, this is selected as the output. If there are tied paths, the scoring strategy is used to select the output among multiple candidate pronunciations. PbA requires a dataset in which each letter of each word's pronunciation is aligned with a corresponding phoneme. The first version of PbA for TTS applications was proposed by Dedina and Nusbaum (1991). They reported the results with a small set of 70 pseudowords at 91% word accuracy. Since then, many studies have used the basic model of Dedina and Nusbaum to improve the PbA system (Sullivan and Damper 1993; Yvon 1996; Damper and Eastmond 1997; Bagshaw 1998; Marchand and Damper 2000; Damper and Marchand 2006).

There are two basic versions of PbA: *explicit* and *implicit* analogy (Damper and Eastmond 1997). Explicit analogy corresponds to *lazy learning* in terms of machine learning, in which the lexicon is retained and the extent of any prior training process is minimised. Implicit analogy corresponds to *eager learning*, in which significant prior training is necessary. For example, the version of Sullivan and Damper (1993) and Bagshaw (1998) are implicit analogy in which the lexicon can be discarded after pre-compiling to produce a knowledge base for generating pronunciation. The other PbA versions in Figure 2.5

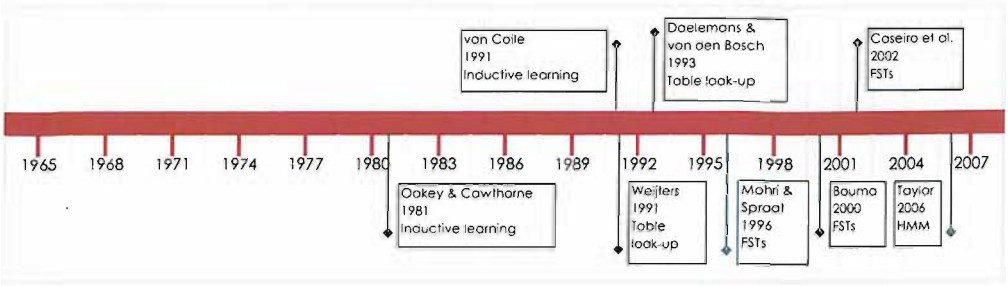

FIGURE 2.6: Historical timeline of other machine learning methods in text-to-phoneme conversion.

are explicit analogy in which the training process is not required and the lexicon is always available at any time. In 1993, Golding and Rosenbloom proposed an analogy-based pronunciation system, called Anapron, and compared it with the other seven pronunciation systems on a surname dataset (see more detail in Section 2.5). In the latest work, Damper and Marchand (2006) proposed a PbA with several further enhancements, boasting a significant improvement in results with three different-size dictionaries. The detail of the PbA approach will be described in the next chapter.

2.4.4 Miscellaneous

The very first data-driven method applied to text-to-phoneme conversion was rule induction (Oakey and Cawthorn 1981). Rules were extracted by using examples of aligned pairs of orthographic string and phonemic form, beginning with a set of 26 rules for each letter pronunciation. Then, the system used this rule set to hypothesize a pronunciation by comparing with a dictionary pronunciation. If the pronunciation was incorrect, a new rule was created to correct the errors. The difficulties in this technique lay in handling the conflict between rules as well as the induction of rules from scratch (Yvon 1994). The work of van Coile (1991) is another example of this technique. The decision tree method may also viewed as a rule induction by some researchers.

After the emergence of NETtalk, the first table look-up method was proposed by Weijters (1991) who drew the conclusion that a simple look-up procedure was superior to a neural network method. The table look-up method can be classified as implicit analogy, in which the lexical database is pre-compiled to yield the generalised phonological knowledge so the dictionary can be discarded. By constructing a table from a training lexicon, the lexicon was compressed into the table containing patterns of letters together with a target phoneme. The patterns of letters in the table have to be accounted for by expanding the context letter to the right or the left. The lack of generalisation power of Weijters's method led van den Bosch and Daelemans (1993) to devise a look-up table with defaults, which could be invoked in the case of a matching failure. The results of both table look-up methods by Weijters and van den Bosch and Daelemans were

reported to have a higher percentage of phonemes correct than those from NETtalk.

Since hand-crafted rule-based approaches have existed for many years, there have been some attempts to improve the system by compiling the rules into finite state transducers (FSTs). FSTs are compact representations that enable one to build models in speech processing more efficiently. One of the advantages of FST representation is that different methods can be combined (Caseiro et al. 2002). The FST-based techniques for text-to-phoneme conversion have been proposed by many researchers for different languages such as English, Dutch and Portuguese, as shown in Figure 2.6. Note that some work in FSTs can be viewed as a rule-based method with a rule compiler.

Hidden Markov models (HMMs) have been applied to the problems in speech processing over the past decades, especially in speech recognition. Recently, they were applied to grapheme-to-phoneme conversion problem (Taylor 2005). In HMMs, phonemes are the hidden states and graphemes are the observations. In Taylor's work, he proposed HMMs and enhancements using pre-processing, a context dependent model and a syllable stress model. The advantage of his work is that the model performs in just one step, including the alignment between graphemes and phonemes. The best result achieved was 61% words correct by using 4-gram with stress adjustments on the Unisyn dictionary (Fitt 2000) containing 110,000 words. This result showed that HMM-based technique can be seen as a competitive method to the table look-up and neural network approach.

The chronological history of miscellaneous methods in this section is shown in Figure 2.6.

2.5 Pronunciation of Proper Names

TTS systems encounter person names, addresses, places, and company names, so the problem of proper name pronunciation has recently received more attention. It is a special problem, because the origin of names may be from many different languages; but they are written in English, which has an extremely irregular spelling system. Many speech synthesisers often rely heavily upon dictionaries. However, creating a dictionary that covers all names is impracticable, despite the availability of large dictionaries and computer memories. Also, the coverage of names can never be 100% by the use of a dictionary because there will always be some rare/new names created in the course of time and those names are not yet included in a dictionary. Thus, back-up strategies must always be provided. This is why proper name pronunciation still presents a challenge for many researchers in speech technology. The most relevant studies for this problem are reviewed in this section.

As an early study in proper name pronunciation, four speech synthesisers that were available in 1984 were evaluated with the 2,000 most common American surnames by Spiegel (1985). The outputs from the four synthesisers were tape recorded, then listened

to and judged by humans. The outputs were rated into four categories: correct, mild error (somewhat acceptably wrong), worst error (embarrassingly wrong), and perceptually wrong due to poor synthesis. Errors were lumped into the last three categories. The author's choice of error criterion was to accept any plausible pronunciation, based on interviews with 3-4 co-workers about ambiguous surnames. As Spiegel admitted, the results were a somewhat *liberal* acceptance of pronunciation. The results for each synthesiser were coded in alphabetic letters (A through D), because the author did not want these as an endorsement for any particular synthesiser. The average error rate for the four synthesisers was 28.7%. The lowest error score was 24.2% and the highest score was 32.8%. The majority of errors were serious (embarrassingly wrong) and occurred on multiple-syllable surnames. The errors made by the synthesisers were mostly unlikely to be made by humans. The author suggested that providing a special set of hand-crafted rules with a large exception phonetic dictionary for names should be a more efficient long-term approach to the problem of surname pronunciation.

With the rule-based method, there has been an assumption that the pronunciation of proper names is difficult to predict using the same pronunciation rules as ordinary words, because it depends on the origin of the spelling (Church 1985; Dutoit 1997, p.125; Holmes and Holmes 2001, p.101). Thus, the TTS system requires special rules for names. This assumption has inspired the applications of LTS rules within language origin classes in the work of Church (1985), Vitale (1991) and Font Llitjós (2001). Church (1985) initially proposed letter-to-phoneme rules for proper names by using the statistics to estimate the language family before applying the specific rules for that language. The statistics were based on the number of occurrences of three-letter sequences in each languages. The performance was claimed to be superior to that of the other rule-based systems for proper names. In the work of Vitale (1991), the name pronunciation system was described and evaluated. Starting from searching in a dictionary, if a name was not found, the language of origin of the name was identified from its characteristic letter patterns. After identifying the language by filter rules, a set of hand-crafted LTS rules for that language was applied to predict the pronunciation. A decade later, this scheme is followed by the work of Font Llitjós (2001) based on the CART technique of Black et al. (1998). Her work aimed to improve pronunciation accuracy of proper names with language origin classes. However this method has quite a low accuracy, reported at 55.22% words correct with stress and 60.76% words correct without stress on a lexicon containing 56,000 names, of which 90% were used for training and the remaining for testing. Consequently, a fully automatic language identifier as implemented in this work is not a propitious answer.

A comparison of Anapron, an analogical pronunciation system, with seven other name pronunciation systems was reported by Golding and Rosenbloom (1993). In this work, eight systems were compared in a subjective aspect, in which listeners rated synthesised pronunciations using a 3-point scale: clearly acceptable, somewhat between, and clearly

bad. The eight systems included Anapron, three early 90s state-of-the-art commercial systems, two variants of NETtalk, and two humans. Anapron generated a rough pronunciation by applying a set of rules adapted from MITalk and elementary textbooks of four European languages, and drawing analogies from names in a case library of 5,000 as exception coverage. The experiment was conducted on 4,000 names, carefully chosen from over 1.5 million surnames. The pronunciations of Anapron and the other systems that have no phonetics-to-speech component were piped through DECtalk and 14 subjects judged the acceptability. The scores of clearly acceptable and somewhat between were lumped together into *acceptable* scores, counting clearly bad as *unacceptable*. The results were given in terms of the percentage of acceptable scores, and showed that the humans performed the best at an accuracy of 93%, Anapron performed at 86%, which was superior to the two versions of NETtalk, but was inferior to the commercial systems.

From a multilingual perspective, there has been a European collaboration in the form of the ONOMASTICA project (The Onomastica Consortium). This project's aim was to create a multi-language pronunciation dictionary of proper names, covering the 11 major languages in European countries: English, Danish, Dutch, French, German, Greek, Italian, Norwegian, Portuguese, Spanish and Swedish. The lexicon consists of different numbers of entries in each language, ranging from 1,000 to more than 1 million names per language. The other objective of this project was the development of semi-automatic pronunciation of proper names. Many techniques were studied depending on each language, such as rule-based methods, neural networks, table look-up methods, and analogy-based approaches. These proposed methods, and the problems in this project, were addressed by Trancoso (1995). The result of this project was a multilingual dictionary of European proper names, which can be used in a dictionary look-up name pronunciation system. One part of this dictionary, which contains 4.5 million entries and has been checked by human experts working on the project, was available on CD-ROM. The fuller lexicon of 8.5 million names was available on magnetic tape. The CD-ROM and tape were distributed to 22 organisations throughout Europe.

Deshmukh et al. (1997) presented an N-best pronunciations system based on a Boltzmann machine neural network that generates the N most-likely pronunciations of surnames from their spellings. A pronunciation dictionary of surnames was created consisting of 18,494 surnames from a diversity of ethnic origins and 25,648 corresponding multiple pronunciations. The Worldbet standards (Hieronymus 1994) were used to perform the phonetic transcription manually. The data were automatically aligned by a dynamic programming algorithm to produce a one-to-one alignment of the spellings with the corresponding phoneme, before using them as the input to the Boltzmann machine. The experimental results with the full data set (15,000 names for training, 3,494 for testing) showed that their method produced a very poor performance. The best performance was 29.33% for all correct pronunciations (counting all of the possible pronunciations as correct) with 200 hidden neurons and a context length of three. They

concluded that their training strategy failed to learn the letter-to-phoneme distribution with conflicting constraints in the training data, regardless of the network architecture. Thus, the basic neural networks were found to be incapable of generating the N-best list pronunciations of proper nouns.

Spiegel (2003) described a large hand-crafted rule-based approach, complemented by a small dictionary. It was claimed that this 15-year research can achieve a high accuracy for a name pronunciation system. The result of this work was a program called Namepro, a pronunciation component in the Orator II TTS system. To generate a pronunciation of an input word, a small exception dictionary was searched. The dictionary was refined using the morphology of names, words, and business neologisms, to contain a couple of dozen entries. If the word was not matched, then it will pass through the steps of ethnographic classification, morphological analysis, LTS rules, syllabification, stress assignment and intonation rules. The model attempts to produce the actual US pronunciations. However, this paper did not present an evaluation of the approach.

A method of deriving rewrite rules from an existing name pronunciation dictionary was proposed recently by Fackrell and Skut (2004). This method has contributed to the dictionary coverage of proper names, rather than a high accuracy for prediction of out-of-vocabulary words. The algorithm was simply based on rule induction from a reverse dictionary, which eliminates the one-to-one mapping of pronunciation and its spelling. The remaining pairs of spellings which share a pronunciation were used to generate an ordered set of rules. As a result, this rule set was able to improve dictionary coverage for surnames by adding 5,000 new entries that corresponds to about half a million names. A subjective evaluation showed that about 80% of the suggested pronunciations are good, with a high degree of agreement among five subjects.

In summary, most researchers tend to develop hand-crafted rules or learning algorithms to cope with name pronunciations. However, high accuracy have not been achieved so far. One obstructive problem encountered is that most dictionaries for proper names have been developed privately and deemed proprietary. Standard dictionaries of proper names are not widely available.

2.6 Conclusion

The conversion of letters into phonemes is not simply a one-to-one mapping process in many languages, for example, English and French which are notorious for their irregularities between the writing system and the phoneme system. Dutoit (1997, p.106) explained that "This phenomenon partly originates in the natural delay between the spoken language, in perpetual evolution, and the much more rigid written one." The most frequently striking example is the substring *ough*, which is pronounced /oʊ/ in the word *although*, /u/ in the word *through*, and /ʌf/ in the word *enough*; the left context

is a key to determine a pronunciation of this substring in each case. This illustrates the lack of invariance in the correspondence between letters and phonemes, and also between the number of letters and the number of phonemes in English words. In many languages, pronunciation of a word may also depend upon lexical features that have no direct manifestation in the spelling of the word (Sproat 1997, p.84). There are some words that have multiple pronunciations such as *read*, which is pronounced differently when it is used in past and present tense. Therefore, sometimes it is extremely difficult to convert from text into phonemes by using only the spelling of word. Furthermore, pronunciation of a word also varies from one person to another, and/or from one moment to another (Dutoit 1997, p.111) especially for proper names; therefore, it is difficult to choose one as an arbitrarily correct answer. Consequently, these make text-to-phoneme conversion a hard problem.

Many attempts have been made to deal with the problem of text-to-phoneme conversion over the years. The first and oldest technique is to develop a rule set by hand, based on linguistic knowledge. The second approach is to use a dictionary to provide transcriptions, a method currently used as a primary strategy because of the accuracy it allows. There are drawbacks to the rule-based approach, such as language dependency, the length of time for rule development, the requirement of explicit knowledge from a human expert, and the poor performance. Also, it is not possible to store all of the words in one language into a system dictionary. There will always be proper names and new words which will be created in the course of time. These have led to the emergence of a backup strategy, a so-called 'data-driven' approach in which it is learned automatically from data.

Inductive learning was the first data-driven method to extract rules from examples of data. However, text-to-phoneme conversion has become a benchmark problem in machine learning since the pioneering work of Sejnowski and Rosenberg on NETtalk in 1987. A large number of different machine learning methods have been applied to the task of automatic transcription, such as various neural networks, decision trees and pronunciation by analogy. All data-driven techniques in the current review are classified as supervised learning, in which the training data consist of pairs of input letters and target output phonemes. These techniques often require alignment between graphemes and phonemes.

Some studies compared different methods. For example, Damper et al. (1999) evaluated the performance of a rule-based approach and three data-driven techniques: PbA, NETspeak, and IGtree. The performance was evaluated with an English lexicon, and reported in terms of words correct which is more stringent and sensitive than phonemes correct. The best performance is obtained with PbA, at approximately 72% words correct on 16,280 words from the Teachers' Word Book dictionary. Recently, Damper and Marchand (2006) reported the best results of PbA at about 87% on 178,041 words from the British English Example Pronunciation dictionary. Therefore, PbA has been

investigated in many useful aspects for using in speech synthesis applications in this thesis.

The problem of automatic pronunciation of proper names has also received considerable attention from researchers. Many researchers attempt to cope with this problem by using traditional approach such as hand-crafted rule-based system and dictionary look-up. Few data-driven approaches have been studied. Some studies augmented their method with languages identification. These pronunciation systems have not yet achieved a high accuracy so far, although promising improvement has been seen. This problem remains a real challenge for text-to-phoneme conversion among speech researchers. Thus, this thesis has been primarily focused on the automatic pronunciation of proper names.

Chapter 3

Pronunciation by Analogy

3.1 Introduction

Pronunciation by analogy (PbA) is a data-driven approach for automatic text-to-phoneme transcription. PbA exploits the phonological knowledge implicitly contained in a dictionary, including words and their pronunciations. The original idea of PbA was based on the theory of reading aloud from Glushko (1979)'s studies. He proposed a psychological model to pronounce pseudowords using analogy with known words which are similar in spelling. This theory has been proved computationally feasible by Dedina and Nusbaum (1991) (D&N). They developed the original and well-known system called PRONOUNCE for TTS applications. It can be categorised into the 'lazy learning' section of the machine learning paradigm, and fits into various groups of the artificial intelligence paradigm such as analogy-based, memory-based, and case-based reasoning, as well as instance-based learning (Damper and Marchand 2006). Since then, there have been many variants of PbA based on the PRONOUNCE system. The variant of PbA implemented to use in this work is based on the classical PRONOUNCE system, with several enhancements as proposed by Marchand and Damper (2000).

In the next section, the principle of the PbA algorithm used in this work is described in detail. Then, previous work on PbA are presented with their results using common word dictionaries.

3.2 Principles

The PbA algorithm consists of three steps: substring matching, building the pronunciation lattice, and a decision function. In substring matching, substrings of the input word are compared with substrings of all words in the lexicon, gaining information of the phoneme set for each matching substring. Then, a directed graph called a *pronunciation*

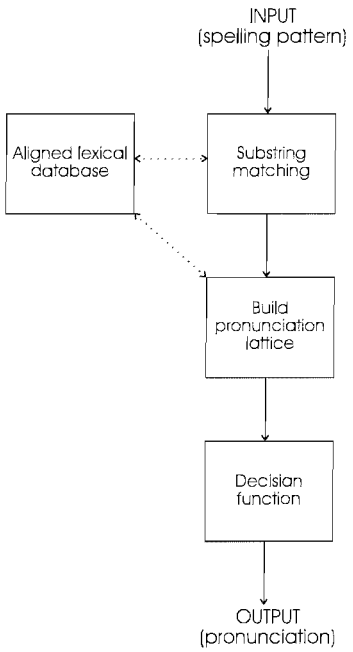

FIGURE 3.1: Dedina and Nusbaum's PRONOUNCE. (Original source: Damper et al. 2001).

lattice is built using information from the previous step. The lattice contains nodes and arcs which represent possible phonemes at particular positions in the pronunciation. Phoneme sets along the paths of the lattice are assembled to determine all possible pronunciations of the input word. In the last step, the decision function will apply heuristic scoring methods to select the best pronunciation from the potential candidates as a final result. The variants of PbA differ mainly in terms of the representative lattice and the heuristic method used in the decision function. The process of PbA is comprised of four components, best described by referring to the original PRONOUNCE program as shown in Figure 3.1. The description given here closely follows Marchand and Damper (2000). This section merely repeats the method for a better illustration.

3.2.1 Aligned Lexical Database

Like most automatic pronunciation techniques, PbA requires a dictionary in which the letters of each word's spelling are aligned with the phonemes of the corresponding pronunciation. In this research, the algorithm presented in Damper et al. (2004) was used to align the lexicon automatically. This alignment used the expectation-maximisation algorithm, which was an iterative approach to a problem where the set of observations (in this case, the pairs of words and pronunciations in the lexicon) was missing some data, and the likelihood function can not be easily differentiated to find its maxima. The missing data were the parameters describing the probabilistic correspondence between words and letters which underlies the alignment process. With this approach, null

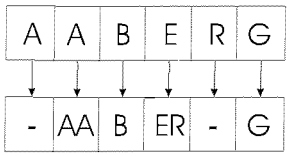

FIGURE 3.2: Alignment of word and its spelling.

symbols (–) were automatically added to the spellings or pronunciations in the lexicon to preserve the one-to-one correspondence. For example, consider a word in the CMUDICT lexicon of spelling AABERG as /AA B ER G/ (CMU phoneme symbols) or /abɜg/ (IPA symbols), for which a possible alignment is shown in Figure 3.2.

After the alignment process, the lexical database was arranged in two columns: the first being the words and the second being the pronunciations (phoneme symbols) to create a sample shown below:

AABERG	–	AA	B	ER	–	G	
AAKER	–	AA	K	ER	–		
AAMODT	–	AA	M	AH	–	T	
AARDEMA	–	AA	R	D	EH	M	AH
AARON	–	EH	R	AH	N		

3.2.2 Substring Matching

When an unknown word is presented as an input to the system, 'full' pattern matching between the input letter string and the dictionary entries is performed. This is called 'full' matching, as opposed to 'partial' matching which was used in the original PRONOUNCE system. It starts with the initial letter of the input string aligned with the end letter of the dictionary entry. If common letters in matching positions in the two substrings are found, their corresponding phonemes and the information about these matching substrings are forwarded to the next module. Then the shorter of the two strings is shifted right by one letter. For partial matching, this continues until the two are right-aligned. For full matching, this continues until the end letter of the input string aligns with the initial letter of the dictionary entry, which means the number of right shifts is equal to the sum of the lengths of the two strings minus one. This process is repeated for all entries in the dictionary. The main reason to use full matching is that the morpheme can be an affix that either comes at the beginning (prefix) or at the ending (suffix), and an affix is often used to form a new word. Thus, as Marchand and Damper (2000) mentioned "full matching seems worth consideration".

To clarify the process, the example of the input word ANN will be given, matching the lexical entry ANNA. Figure 3.3 illustrates the full pattern matching process of this example. At the first iteration, the initial A in ANNA aligns with the final N in ANN, and there are common substrings. Next, ANN and ANNA substrings are shifted one

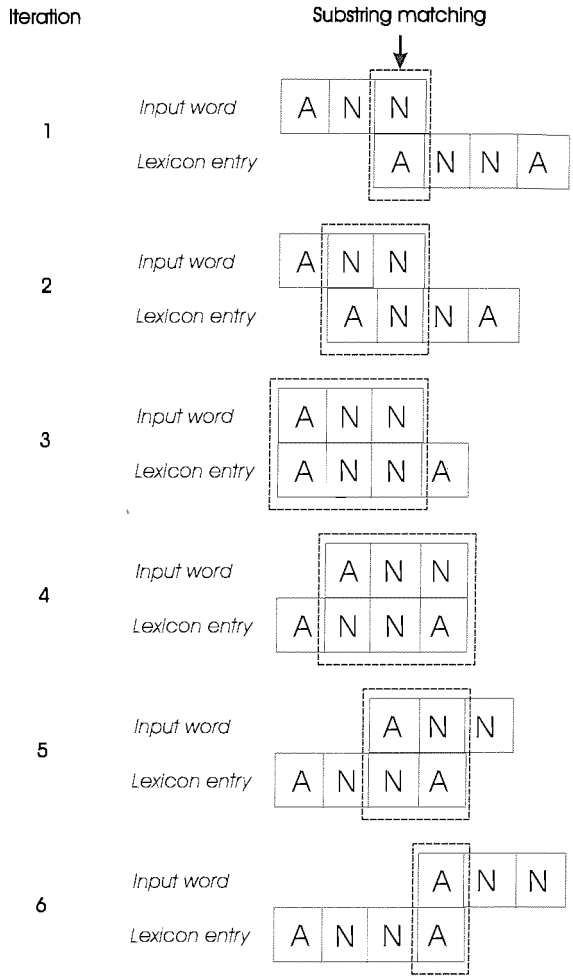

FIGURE 3.3: Substring matching between input word ANN and lexicon entry ANNA.

position, giving no common substrings from the AN in ANNA aligning with the NN in ANN. The third iteration, the common substring ANN, is extracted from the alignment between the input string ANN and the ANN in ANNA. The process terminates at the sixth iteration, when the final A in ANNA aligns with the initial A in ANN, giving the common substring A. For partial matching, the process stops at iteration 4.

3.2.3 Building the Pronunciation Lattice

The pronunciation lattice is a directed graph containing nodes and arcs. Matching substring information is used to construct nodes and arcs in the lattice for the input string. A lattice node represents a matched letter, L_i, at some position, i, in the input. The node is labelled with its position i and the corresponding phoneme to L_i in the matched substring, P_{im} say, for the mth matched substring. An arc is labelled with phonemes between P_{im} and P_{jm} in the relevant part of the matched substring and the frequency count, increasing by one each time the substring with these phonemes is

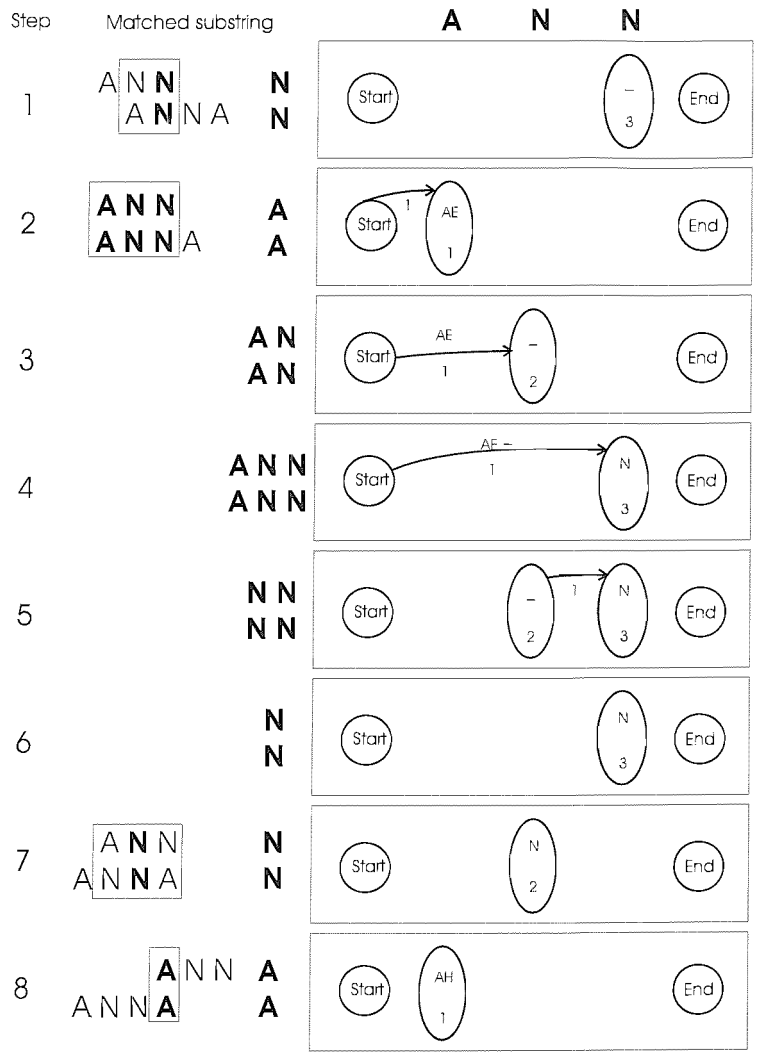

FIGURE 3.4: The step-by-step process to build a pronunciation lattice for the word ANN with the matched substrings from the lexical entry ANNA, corresponding to the pronunciation /AE – N AH/. The matched substrings are shown in bold.

matched during the search throughout the lexical. If the arcs correspond to bigrams (the two phonemes), the arcs are labelled only with the frequency. Bigram phonemes label the nodes at both ends. Additionally, there is a *Start* node at position 0, and an *End* node at position, which is the length of the input string plus one.

A step-by-step example in Figure 3.4 is given to clarify how to build the lattice for the input word ANN with the matched substrings from the lexical entry ANNA, corresponding to the pronunciation /AE – N AH/. Each node and arc in the lattice is constructed by using the lists of matched substrings from the previous module. The explanation for each step is as follows:

Step 1: start with the matched substring N found in iteration 2, the position of N in the input is 3 and the position of N in the lexicon entry is 2, giving the phoneme /–/. With

this information, the node (/–/, 3) is created.

Step 2: the input string ANN is matched with the substring ANN in the word ANNA (iteration 3). The matched substring ANN can be divided into 5 matched substrings: A, AN, ANN, NN, N. For the matched substring A, the node (/AE/, 1) and an arc with label (1) are created, since the position of A in the input and in the lexicon entry are the same, which is the position *Start* giving the phoneme /AE/.

Step 3: for the matched substring AN, first, the node (/–/, 2) because the last letter N in substring AN, is matched at position 2 in the input and lexicon entry giving phoneme /–/. Then, the arc with label (/AE –/, 1) is created to connect from *Start* to the node (/–/, 2) because the first letter A in the matched substring AN, is match at the beginning of the input word and the lexicon entry giving phoneme /AE/.

Step 4: the input string ANN is matched with the substring ANN, giving the phonemes /AE – N/ for the substring from *Start* to 3. With this information, the node (/N/, 3) is created, since the last position of the matched substring is 3 and an arc with label (/AE –/, 1) is created.

Step 5: for the matched substring NN, the nodes corresponding to these 2 letters, N at positions 2 and 3, are already created from the previous step. Thus, the arc with label (1) is created to link between node (/–/, 2) and node (/N/, 3).

Step 6: the node (/N/, 3) corresponding to the matched substring N is already created from the previous step.

Step 7: the matched substring N is found in iteration 4, the position of N in the input is 2 and the position of N in the lexicon entry is 3, giving the phoneme /N/. Thus, the node (/N/, 2) is created.

Step 8: the matched substring A is found in iteration 6, the position of A in the input is 1 and the position of A in the lexicon entry is 4, giving the phoneme /AH/. Thus, the node (/AH/, 1) is created.

Figure 3.5 represents the final pronunciation lattice corresponding to the input word, ANN, given the following words in the lexicon: (ANNA, /AE – N AH/), (AN, /AE N/), (AND, /AE N D/), and (AMANN, /AE M AH – N/). In this example, there are two shortest paths and both paths produce the correct pronunciation as /AE – N/.

3.2.4 Decision Function

Finally, the decision function finds the complete shortest path through the lattice from *Start* to *End*. A justification for using only the shortest path has not been directly stated by D&N; one possibility is that to find all paths is time-consuming, therefore considering only the shortest paths would reduce the computational time. Furthermore,

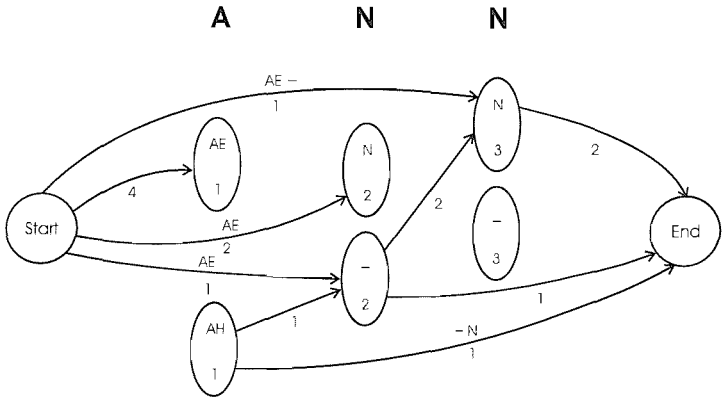

FIGURE 3.5: The pronunciation lattice for the word ANN.

pronunciation errors tend to appear with more frequency at the boundaries of matched substrings than at the inner letters, and so reducing the number of arcs in the lattice tends to reduce the chance of error. The possible pronunciation for the input corresponds to the output string assembled by concatenating the phoneme labels on the nodes or arcs in the order that they are traversed. In the case of only one candidate corresponding to a unique shortest path, this is selected as the output. If there are tied shortest paths, multiple scoring strategies are used to select the output. Note that, different paths can correspond to the same pronunciation.

There are five different strategies used by Marchand and Damper (2000) in calculating scores for all pronunciation candidates. Scores are then ranked in either ascending or descending order. These ranks are used to give points for the candidates on each strategy. These points are next multiplied together to get the final score.

The number of points given to each scoring strategy depends on the number of candidate pronunciations. The total numbers of points *(T)* awarded for each strategy is:

$$T(N) = \sum_{r=1}^{N} r = \frac{N(N+1)}{2} \tag{3.1}$$

where N is the number of candidate pronunciations.

These T points are divided among the candidates depending on the rank of each one. Let cand(R_{S_i}) equal the number of candidates with rank R for the scoring strategy S_i, then $P(C_j, R_{S_i})$, the number of points awarded to candidate C_j is:

$$P(C_j, R_{S_i}) = \frac{\sum_{i=R_{S_i}}^{R_{S_i}+\text{cand}(R_{S_i})-1}(N-i+1)}{\text{cand}(R_{S_i})} \tag{3.2}$$

The final score for each candidate, $FS(C_j)$, is simply calculated as the product of the

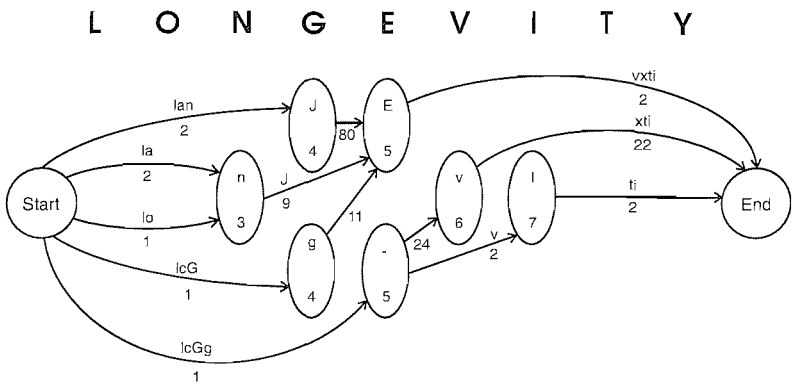

FIGURE 3.6: Partial pronunciation lattice for the word LONGEVITY. (Original source: Marchand and Damper 2000).

numbers of points yielded from each of the S strategies as follows:

$$FS(C_j) = \prod_{i=1}^{S} \delta_{S_i} P(C_j, R_{S_i}) + (1 - \delta_{S_i}) \tag{3.3}$$

where δ_{S_i} is 1 if strategy S_i is included in the combined score, and 0 otherwise.

Finally, the result is the pronunciation candidate that achieved the highest final score.

Scoring strategies

To explain each strategy, the representation of the sets used in the lattice is as follows:

$L(W_i) = C_1,...,C_j,...,C_N$ is the lattice for the word W_i.

$C_{j\in[1,N]}$ denoting the candidates and C_j is consisted of $(F_j,\ D_j,\ P_j)$ where:

> $F_j = f_1,...,f_n$ is the set of arc frequencies along the jth candidate path (length n).
>
> $D_j = d_1,...,d_k,....,d_n$ is the set of the difference of the position index of the nodes at either end of the kth arc, called the path structure.
>
> $P_j = p_1,...,p_m,....,p_l$ is the set of pronunciation candidates with p_m's from the set of phonemes and l is the length of the pronunciation.

To clarify how to calculate the scoring method for each strategy, the pronunciation lattice given in Figure 3.6 is used as an example. Note that the NETtalk phoneme symbols are used in this example. From this lattice, the program produces the six candidate pronunciations as presented in Table 3.1.

Strategy 1: The product of the arc frequencies *(PF)*. The candidate scoring values are ranked in descending order.

Candidate	Pronunciation	Path structure	Arc frequencies
1	/lanJEvxti/	{4, 1, 5}	{2, 80, 2}
2	/lanJEvxti/	{3, 2, 5 }	{2, 9, 2}
3	/lonJEvxti/	{3, 2, 5}	{1, 9, 2}
4	/lcGgEvxti/	{4, 1, 5}	{1, 11, 2}
5	/lcGg–vxti/	{5, 1, 4}	{1, 24, 22}
6	/lcGg–vIti/	{5, 2, 3}	{1, 2, 2}

TABLE 3.1: The six candidate pronunciations for the word LONGEVITY.

$$PF(C_j) = \prod_{i=1}^{n} f_i \tag{3.4}$$

Candidate	1	2	3	4	5	6
Score, *PF()*	320	36	18	22	528	4
Rank	2	3	5	4	1	6
Points	5	4	2	3	6	1

TABLE 3.2: The computation of *PF()* for the six candidates.

Strategy 2: the standard deviation of the values associated with the path structure *(SDPS)*. The candidate scoring values are ranked in ascending order.

$$SDPS(C_j) = \sqrt{\frac{\sum_{i=1}^{n}(d_i - \bar{d})^2}{n}}, \quad \bar{d} = \frac{\sum_{i=1}^{n} d_i}{n} \tag{3.5}$$

Candidate	1	2	3	4	5	6
Score, *SDPS()*	1.7	1.2	1.2	1.7	1.7	1.2
Rank	4	1	1	4	4	1
Points	2	5	2	2	2	5

TABLE 3.3: The computation of *SDPS()* for the six candidates.

Strategy 3: the frequency of the same pronunciation *(FSP)*, i.e. the number of occurrences of the same pronunciation within the tied shortest paths. The candidate scoring values are ranked in descending order.

$$FSP(C_j) = \text{cand}\{P_j | P_j = P_k\} \quad with \ j \neq k \quad and \ k \in [1, N] \tag{3.6}$$

Candidate	1	2	3	4	5	6
Score, *FSP()*	2	2	1	1	1	1
Rank	1	1	3	3	3	3
Points	5.5	5.5	2.5	2.5	2.5	2.5

TABLE 3.4: The computation of *FSP()* for the six candidates.

Strategy 4: the number of different symbols *NDS()* between a pronunciation candidate C_j and the other candidates. The candidate scoring values are ranked in ascending order.

$$NDS(C_j) = \sum_{i=1}^{n} \sum_{k=1}^{N} \delta(P_{j,i}, P_{k,i}) \tag{3.7}$$

where $\delta()$ is 1 if pronunciations P_j and P_k differ in position i and is 0 otherwise. Table 3.6 illustrates the computation of *NDS()* for Candidate 1.

Candidate	1	2	3	4	5	6
Score, *NDS()*	13	13	14	12	14	18
Rank	2	2	3	1	3	6
Points	4.5	4.5	2.5	6	2.5	1

TABLE 3.5: The computation of *NDS()* for the six candidates.

Candidate 1	l	a	n	J	E	v	x	t	i
Other candidates to be compared with Candidate 1	l	a	n	J	E	v	x	t	i
	l	**o**	n	J	E	v	x	t	i
	l	**c**	**G**	**g**	E	v	x	t	i
	l	**c**	**G**	**g**	–	v	x	t	i
	l	**c**	**G**	**g**	–	v	**I**	t	i
Differences at position i	0	4	3	3	2	0	1	0	0
NDS() Score	13								

TABLE 3.6: Illustration of the computation of *NDS()* for Candidate 1. Phonemes differed to those of the target pronunciation are written in bold.

Strategy 5: weak link *WL()*, i.e.the minimum of the arc frequencies. The candidate scoring values are ranked in descending order.

$$WL(C_j) = \min_i f_i, \quad i \in [i, n] \tag{3.8}$$

Candidate	1	2	3	4	5	6
Score, *WL()*	2	2	1	1	1	1
Rank	1	1	3	3	3	3
Points	5.5	5.5	2.5	2.5	2.5	2.5

TABLE 3.7: The computation of *WL()* for the six candidates.

Final result: these five strategies for scoring the shortest paths are used to determine the final result by combining all possible combinations. The number of possible combinations is $(2^5 - 1) = 31$. A 5-bit code is represented by the combinations in which '1' at position i indicates that strategy S_i was included in the combination, or otherwise '0'. For example, the code '10000' indicates that strategy 3 *(FSP)* is used singly, and

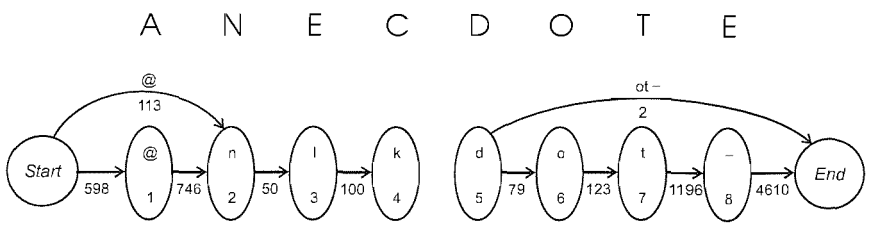

FIGURE 3.7: The silence problem occurs in the word ANECDOTE which fails to produce an output. (Original source: Damper and Eastmond 1997).

the code '10101' indicates that strategies 1, 3, and 5 are used together. An example of the use of combination '10101' for deriving a pronunciation of the word LONGEVITY is shown in Table 3.8. The points used to calculate the final score are shown in bold. As can be seen, the final result is Candidate 1 which is the correct pronunciation in this case.

Candidate	**PF**	SDPS	**FSP**	NDS	**WL**	Final score
1	**5**	2	**5.5**	4.5	**5.5**	151.25
2	**4**	5	**5.5**	4.5	**5.5**	121
3	**2**	5	**2.5**	2.5	**2.5**	12.5
4	**3**	2	**2.5**	6	**2.5**	18.75
5	**6**	2	**2.5**	2.5	**2.5**	37.5
6	**1**	5	**2.5**	1	**2.5**	6.25

TABLE 3.8: Example of multi-strategy scoring for the word LONGEVITY using the '10101' combination and product rule.

One drawback of this implementation is that it suffers the silence problem, which occurs when there is no complete path from *Start* to *End* nodes. In this case, the program can not produce any pronunciation. An example of the silence problem is shown in Figure 3.7. In the pronunciation lattice of the word ANECDOTE, the figure shows only a subset of the arcs for clarity. This lattice has no arc between nodes (/k/, 4) and (/d/, 5) because there is no substring *cd* matching with phonemes /kd/ in the dictionary except in the word itself. Thus, there is no complete path through this lattice, so the program fails to produce the pronunciation of this word. As a result of the silence problem, an output is not necessarily guaranteed using PbA alone.

3.3 Appraisal with Common Words

In this section, previous work is briefly reviewed with their results mostly on lexicons of common words. The main differences of each variant of PbA are addressed.

3.3.1 PRONOUNCE

In the work of D&N, the classical PRONOUNCE program was evaluated on 70 monosyllabic pseudowords, a subset of those previously used in reading studies by Glushko (1979). Approximately 20,000 words from Webster's Pocket Dictionary were used as a lexicon of known words. In substring matching, they used 'partial' matching which starts with the left-most letter of the input word and of the dictionary entry, and continues until the two are right-aligned. In the decision function, if there are tied shortest paths, the sum of arc frequencies of each candidate is computed. The output is the pronunciation obtained with the highest score. This is similar to strategy 1 in the multiple-strategy approach, in which the product of arc frequencies has been used instead. The correct pronunciations of these pseudowords were given by seven human subjects. They reported results, in terms of word accuracy, at 91%. However, their test set was not representative of general English. Later work showed repeating of the experiment on both sets of pseudowords and sets of realistic words failed to accomplish such a high accuracy (Sullivan and Damper 1993; Yvon 1996; Damper and Eastmond 1997; Bagshaw 1998).

3.3.2 Synthesis-by-Analogy

The impact of implementational choices on performance of the PbA system has been studied by Sullivan and Damper (1993). They developed a synthesis-by-analogy system by employing analogy in both orthographic and phonological domains. Three different English lexicons and one German database was used with different treatment of word boundaries and candidate scoring methods. Each lexicon consisted of 800 words except one English lexicon which consisted of 3,926 words. The pronunciation lattice was different from the D&N model, in which the nodes represent the junctures between letters or phonemes rather than the letters or phonemes themselves, and the arcs represent possible phoneme(s) with preference values. The preference value reflects the probability of individual orthographic-to-phoneme mappings. Strategy 1 was used as a scoring method. In evaluating the system's output, a set of pseudowords was created and the 'correct' pronunciations were given by native speakers (131 words in English and 100 words for German). A pseudoword may have more than one 'correct' pronunciation, that is the pronunciation produced by any subject was considered as a correct pronunciation. The best results for the two languages were obtained from different implementations. For English, phonemic analogy performed much better than orthographic analogy. However, the reverse was true for German. The best performance for English and German was 78.7% and 82% words correct respectively. However, a high accuracy in results was not achieved for English; in fact, the error rates on their similar test set were much higher than PRONOUNCE.

3.3.3 Multiple Unbounded Overlapping Chunks

Yvon (1996) proposed an extension of D&N's algorithm in which the pronunciation lattice has been modified. To construct the lattice, this system used multiple unbounded overlapping portions of existing transcriptions, and a shared phoneme between contiguous matching substrings is applied. In order to find the output, the recombination of the smallest number of nodes that overlap maximally is deemed a possible pronunciation. Then, if there are the possible ties, the best pronunciation corresponding to the path that has the minimum weight in a lattice is chosen as an output. His extension has been evaluated and compared with PRONOUNCE and DEC, a decision tree method by Torkkola (1993). Five dictionaries from different languages were used for evaluation. Two of them are common-word dictionaries: NETtalk (English) and BDLEX (French), and the rest are proper-name dictionaries (Italian, French, Dutch). The results showed that his method outperforms PRONOUNCE in all lexicons. For a decision tree approach, the accuracy of results is in between those of PRONOUNCE and his algorithm. The results of English lexicons were poor (63.96% words correct) when compared to the other languages, and the results of the Italian lexicon achieved the highest word accuracy (95% words correct). Nevertheless, he concluded that his analogy-based models failed to provide appropriate pronunciations for the phonetic transcription task.

3.3.4 Pronunciation by Analogy

Damper and Eastmond (1997) studied the impact of implementational choices on the performance of various PbA models. Three models were evaluated; the first model was the re-implementation of D&N, and the other two models were also based on the D&N's model but with different scoring methods: the maximum product of the arc frequencies (PROD), and the maximum sum of the score from all paths corresponding to the same pronunciation so-called total product (TP). Sets of pseudowords and sets of lexical words were tested. There are two sets of pseudowords: the 70 pseudowords used in PRONOUNCE and the 131 pseudowords used by Sullivan and Damper (1993) plus 5 more words. Also, two databases, Webster's and Teacher's Word Book (TWB) dictionary, were used as a test set and a lexicon. Each lexicon was tested by using the leave-one-out strategy; that is, each word is removed in turn from its lexicon, and a pronunciation inferred by analogy with the remainder. The replication of D&N models was unable to reproduce the same level of performance reported in their original work. The best results were obtained with the TP model in all cases. In the testing of lexical words, results on TWB were better than on Webster's in all models.

3.3.5 Phonemic transcription by Analogy

In the work of Bagshaw (1998), pronunciation by implicit analogy has been proposed for a real-time synthesiser. Context-dependent grapheme-to-phoneme rules were derived from a phonetic dictionary in the training stage. Then, the dictionary can be discarded. These rules were minimised to maintain sufficient speed for real-time processing. Principally, a lexical lookup with a system of pronunciation by analogy was used to generate a pronunciation. A lattice was generated by searching for context-dependent grapheme rules that match at each letter. In the case that no rule was matched, an exception rule was invoked, in which the most probable pronunciation for that letter regardless of its context is given. Each path in a lattice was considered as a possible pronunciation by calculating a score S, which is the sum of weights of the applied rules, and a penalty score Z, which is the number of exception rules applied in a path. In Bagshaw's PbA system, the syllable boundary and stress were also included in the transcriptions. The CNET lexicon of 110,000 words was used as training and testing data. The result achieved was 77.06% words correct. The lexicon of 68,000 proper names from Onomastica was also evaluated and yielded a lower accuracy at about 49%.

3.3.6 Multi-Strategy PbA

More recently, Marchand and Damper (2000) proposed a multi-strategy approach to improving pronunciation by analogy with some simple heuristics for silence avoidance. They also extended the method to the problems of phoneme-to-letter conversion and letter-to-stress conversion. The algorithm of this variant with multiple strategies was already presented in Section 3.2.4. The multiple strategies were studied with two combination methods: the product rule and the sum rule. Webster's dictionary was tested and the best results obtained were 65.5% word accuracy from the product of all five strategies. The results of this multi-strategy approach showed a significant improvement in the performance over the single strategy versions of the PbA. Furthermore, this multi-strategy PbA was compared with the rule-based approach, and two data-driven techniques: NETspeak and IGTree (Damper et al. 1999). The results showed that PbA outperforms the other methods on a small lexicon. The best results were obtained at approximately 72% of words correct on the TWB database. Thus, PbA has been re-implemented for use in this thesis based on the version of Marchand and Damper (2000).

Table 3.9 shows the best results obtained from different versions of PbA, mentioned in this section, with various languages and lexicons.

The work of ...	Lexicon	Number of word	Language	Number of test set	Type of word	% Words correct
Dedina & Nusbaum (1987)	Webster	20,009	English	70	pseudoword	91.00
Sullivan & Damper (1993)	Basic English	800	English	131	pseudoword	78.70
	KF800	800	English	131		62.50
	OALD	3,926	English	131		74.00
	Meier	800	German	100		82.00
Yvon (1996)	Webster	20,009	English	the 10th of lexicon	common word	63.96
	BDLEX	≈20K-25K	French		common word	86.54
	IT-NP	≈20K-25K	Italian		proper name	95.73
	FR-NP	≈20K-25K	French		proper name	79.14
	NE-NP	≈20K-25K	Dutch		proper name	89.86
Damper & Eastmond (1997)	Webster	20,009	English	70	pseudoword	85.70
	Webster	20,009		136	pseudoword	85.30
	Webster	20,009		whole lexicon	common word	60.70
	TWB	16,280		whole lexicon	common word	67.90
Bagshaw (1998)	CNET	114,614	English	whole lexicon	common word	77.06
	CNET	114,614		68,046	proper name	49.21
Marchand & Damper (2000)	Webster	19,594	English	whole lexicon	common word	65.50
	TWB	16,280		whole lexicon	common word	71.80

TABLE 3.9: The best results of each version of PbA with various lexical databases.

3.4 Conclusion

The psychological model of Glushko (1979) and the first implementation program by D&N have greatly influenced automatic pronunciation methods by analogy. Since then, pronunciation by analogy has been a well-known and successful approach for phonetic transcription in TTS synthesis. PbA has been studied and extended by many researchers. The principle of PbA used in this thesis has been described. The various extensions of PbA were addressed with their results.

The assumption underlying PbA is that the dictionary contains implicit phonological knowledge which can be exploited to generate a pronunciation for an unknown word. By hypothesising a partial pronunciation from lexicon for each matched substring between input word and lexical entries, an output is formed by joining those partial pronunciations. This idea denies the assumption of Miller (1981, p.49) that:

> "... the pronunciation of part of a word is not generally part of its pronunciation: if x and y are segments of a word, then Pronunciation(x)+ Pronunciation(y) $\neq$ Pronunciation(x+y)."

PbA is a kind of explicit analogy or lazy learning, in which the prior training process is minimised and the dictionary is still kept for looking up or inferring a pronunciation; this may be one of the reasons that makes PbA more successful than other methods. In fact, there is good empirical evidence to support this reason; Daelemans et al. (1999) provided the empirical results suggested that keeping exceptional training instances in memory can be beneficial for generalisation accuracy in language learning. They also suggested that editing or abstracting instances in implicit analogy methods, such as decision-tree learning, can be harmful. Some rare information maybe discarded while compressing the training data into some other kind of representation.

With very good results obtained in terms of word accuracy, PbA seems to be a very promising approach. Marchand and Damper (2000) revealed that the performance of PbA still has room for improvement based on selecting among shortest paths. There are many aspects of PbA which may help to improve performance and have not been studied yet. Furthermore, D&N mentioned that pronunciation by analogy may cope with the problem of surname pronunciation. Consequently, PbA was chosen for further investigation in this thesis and the problem of name pronunciation is focused.

Chapter 4

Syllabification by Analogy of Proper Names

4.1 Introduction

The syllable is an important unit of a word often mentioned in phonological studies. However, its definition is still a controversial issue among the linguistic community (Holmes and Holmes 2001, p.286). There are various definitions for a syllable; here is the example of a definition from the Cambridge Advanced Learner's Dictionary:

> "Syllable [noun] a single unit of speech, either a whole word or one of the parts into which a word can be separated, usually containing a vowel.".

The separation of a word into syllables is called syllabification. In English, it is difficult to syllabify words, whether spoken or written, due to the difficulty in defining a syllable and the irregular correspondence between spelling and sound. The process of syllabification is ambiguous whether it should operate in the orthographic or phonological domain. Strictly speaking, the term syllable might be more accurately applied only to the phonological domain. However, it is often used to apply in the orthographic domain, related to a hyphenation problem since the syllable boundary is usually marked by a hyphen. For instance, the possible syllabification of the word FEELING might be /fi-lɪŋ/ in the pronunciation domain, or alternatively FEEL-ING in the spelling domain.

Irrespective of the definition, syllabification is undoubtedly one of the important component in any TTS system (Kiraz and Möbius 1998). In most languages, syllable boundaries are usually treated as an aid to pronunciation. The syllable is also used to assign the stress on a word and its location affects the duration of the phone (Tian 2004; Marchand and Damper 2007). In unit selection of concatenative synthesis, syllables may be used as a basic unit.

Native speakers of a language are able to count syllables easily based on intuition, but their given boundaries can vary from one to another (Marchand and Damper 2007). Furthermore, there are no rules or algorithms for syllabification that have been generally accepted so far. Many attempts to syllabify have been proposed (Tian 2004). Thus, designing an algorithm for dividing words into syllables is still a challenging problem in a wide range of research fields, not only in the linguistic area but also in machine learning and speech technology.

Different approaches have been used to solve the problem of automatic syllabification and/or hyphenation. Rule-based systems, which are mainly reliant on a maximal onset principle (Pulgram 1970, p.47; Kahn 1976, p.41) or a sonority hierarchy (Clements 1988), are the traditional methods applied to the problem. A few data-driven approaches have been proposed, especially for English and German. Daelemans and van den Bosch (1992) investigated the backpropagation neural network approach for Dutch and compared the results with various symbolic pattern matching approaches and an exemplar-based generalisation technique. The training set consisted of 19,451 words and the test set consisted of 1,945 words. Different neural networks have been tested and the ultimate accuracy achieved was about 96% correctly-placed hyphens. Comparison of performance showed that the backpropagation approach was not superior to the other two methods. The FST method has been applied in the phonological domain for German and English syllabification by Kiraz and Möbius (1998). However, evaluation results have not been formally reported. A probabilistic context-free grammar has also been used to predict syllable boundaries for German (Müller 2001). The corpus was split into 10 folds and various models of grammar were evaluated by training on 9 folds and testing with the rest (approximately 240,000 words). The best model achieved up to 96.4% word accuracy. Tian (2004) investigated two data-driven approaches for modelling syllabification: decision-tree and neural network. Both methods were evaluated on 108,080 words from CMUDICT. However, the number of words used as train and test sets in this paper is not clear. His results showed that when training with 2,000 samples, neural network approach performed better than the decision tree approach. In a recent paper, syllabification was formulated as a tagging task, and a syllable tagger based on an HMM was proposed by Schmid et al. (2007). The German CELEX dictionary, containing about 300,000 words, was evaluated using 10-fold cross-validation. The results shows the syllable boundaries were correctly predicted with a very high accuracy of 99.85%. Recently, Marchand and Damper (2007) introduced a syllabification algorithm for English. The results showed that integrating syllable boundary information manually in the orthographic input can dramatically improve the performance of automatic pronunciation by analogy. They reported that the rate of word accuracy was increased by approximately 5% when the correct (according to the corpus) positions of syllable boundaries were given. However, the automatic syllabification that they proposed using the same concept as PbA did not yield better results.

In this chapter, the aim is to assess the possibility of improving the performance of PbA by using the same analogy concept to syllabify proper names. Experiments were carried out using the concept of syllabification by analogy (SbA) proposed by Marchand and Damper (2007). SbA is an algorithm for determining syllable boundaries in the orthographic form by using analogical reasoning from a lexicon of known syllabifications. The experiments were conducted using a separate syllabification step, and then inferring the pronunciation by analogy. In the remainder of this chapter, the syllable structure of the dictionary used in this task is given. The SbA algorithm is described next. Then the implementation and results are presented when SbA is applied to the problem of proper names. Finally, discussion and conclusion are presented in the last section.

4.2 Syllable Structure

Typically a syllable consists of one or more vowel sounds preceding or following zero/-more consonant sounds or certain consonants alone. This structure generally applies in the phonological domain, not the orthographic domain. However, the SbA algorithm must operate in the orthographic domain, since its input is text. Also, there is no syllable boundary in a dictionary of proper names. It is interesting to see how syllabification reflects into the orthographic domain. Thus, in this experiment the syllable boundaries are applied in the spelling domain as below.

Webster's dictionary is the only dictionary, from all dictionaries used in this thesis, that contains pronunciation with a stress and syllable boundary for each word. Thus, this lexicon was used for inferring the syllabification, and a dictionary of proper names was used for inferring the pronunciation. A sample of Webster's dictionary is shown as below:

aardvark	a-rdvark	1 <<<> 2 <<
aback	xb@k-	0 > 1 <<
abacus	@bxkxs	1 < 0 > 0 <
abaft	xb@ft	0 > 1 <<

The lexical database is arranged in three columns: the spelling, the pronunciation (expressed in the phoneme symbols listed in Appendix B), and the stress and syllabification patterns. The syllable boundaries can be extracted from the third column by inferring from four regular expressions:

R1:	[<>]	⇒	[< | >]
R2:	[<digit]	⇒	[< |digit]
R2:	[digit>]	⇒	[digit| >]
R2:	[digit digit]	⇒	[digit|digit]

The number in the third column indicates the level of stress: 1 for primary stress, 2 for secondary stress, and 0 for tertiary stress. The '<' and '>' symbols denotes the right

and left syllable boundary. The '|' symbol in the rules denotes the syllable boundaries. For the above example words, after applying these rules, the syllabifications are:

<AARD|VARK>
<A|BACK>
<AB|A|CUS>
<A|BAFT>

4.3 Syllabification by Analogy

Syllabification by analogy is a modification of PbA which predicts syllable boundaries from their spelling by inference from a syllabified dictionary. It was initially proposed by Marchand and Damper (2007). They developed a syllabification system by employing analogy in the orthographic domain. The following description of SbA and three models of syllabification/pronunciation in this section are elaborated in their article.

4.3.1 Principle of SbA

The principle of SbA is similar to that of PbA in Chapter 3. The main difference is that the junctures between letters are added explicitly to modify PbA in three parts: input words, lexical entries, and the output. The '*' symbols represent possible boundaries in the input words, and are used as input symbols to label the lattice nodes, for example, <ABAFT> is expanded to <A*B*A*F*T>. For lexical entries, the '|' symbols indicate the positions of syllable boundary and '*' is also added as a juncture where there is no boundary presence, e.g., <A|B*A*C*K>. Both '*' and '|' are represented as possible output symbols, and are used to label the arcs in a lattice. In substring matching, '*' in the input can match with either '*' or '|' in lexical entries. A '*–*' match is entered in the lattice as a '*' and a '*–|' match is entered in the lattice as '|'. The following processes are performed in exactly the same way as in PbA, except that in the final step, the '*' symbols are removed to yield the output. One problem with SbA is that the processing time increases significantly compared to PbA, due to the extension of word lengths by juncture symbols.

4.3.2 Three Models of Syllabification and Pronunciation

The concept of SbA has been adapted to infer syllabification and pronunciation together. This was done to investigate whether automatic syllabification can lead to superior pronunciation performance or not. There are three models:

- A perfect model, S*(P)bA – in this case, we assume that the input word is already syllabified correctly and compared to syllabified lexical entries, giving matched

substring with their corresponding pronunciations including syllable markers to construct the lattice. From here, the process proceeds exactly as for PbA.

- A parallel model, (S||P)bA – the syllabification of an input word is unknown, and the process is similar to SbA, except that pronunciation information and syllable markers are included to build the lattice. From here, the process proceeds exactly as for PbA.
- A series model, (S+P)bA – the syllabification of the input is given by processing SbA first, then this is used as an input for processing as in the S*(P)bA model.

4.3.3 Previous Results

The models in the previous sections were evaluated using 19,596 entries from Webster's dictionary – the 413 homonyms were removed from the original NETtalk. A fair test was conducted by using the leave-one-out strategy and reported in terms of words correct. The best results were obtained from SbA using the 10101 combination at 78.10% word correct and 93.1% boundaries correct. For the three models for inferring syllabification and pronunciation together, the results were reported in terms of word accuracy, which means if all phonemes of a word, including the null phoneme, are correct then this word was counts as a correct pronunciation. The best results of S*(P)bA achieved a very significant improvement compared to those of PbA, from 65.35% words correct for PbA to 71.74% words correct for S*(P)bA. This illustrated that perfect syllabification successfully improved the performance of pronunciation. However, the other two models failed to improve the performance of PbA. The results of a parallel model were slightly superior to those of the series model. They both obtained their best results from the 10100 combination, 64.82% words correct for the (S||P)bA model and 64.26% words correct for the (S+P)bA model.

4.4 Experimental Results

Because of the success in improving the performance of pronunciation using a perfect model for inferring syllabification and pronunciation, an investigation was carried out to see how well automatic syllabification can cope with proper names. Experiments were carried out using the (S+P)bA model with the 52,911 proper names in CMUDICT. The set of proper names in CMUDICT was syllabified by inferring the syllable boundaries from a dictionary of common words (Webster's dictionary). The details of these two dictionaries are fully described in the next chapter. At this stage, the percentage of boundaries correct can not be determined because the correct syllabifications of these proper names were unavailable. Thus, the different outputs were created from all combinations as we do not know which combination can produce the best syllabification

(close to 'perfect'). Then these were used as an input for the next step, the normal PbA, to find the pronunciation. The results were evaluated using the leave-one-out method. Tables 4.1–4.4 show the results of syllabification and pronunciation by analogy, using the (S+P)bA model with CMUDICT proper names in the form of the confusion matrix for all combinations of SbA and (S+P)bA. The best results are written in bold. As can be seen, the best results were achieved when syllabified using the 00010 combination, and when the pronunciation was inferred using the 10101 combination. However, the best result of 65.14% words correct for (S+P)bA versus 68.35% for PbA was significantly poorer (binomial tests, one-tailed, $p << 0.01$). The best performance was observed using a 10101 combination of (S+P)bA.

The (S||P)bA model can not be evaluated because Webster and CMUDICT use different phoneme inventories. Therefore, syllable boundaries and pronunciation can not be inferred at the same time.

4.5 Conclusion

Syllabification is an important task in speech recognition and synthesis, since syllables can help a pronunciation system in assigning stress and duration of phones. Recent work showed that manually-syllabified information enhanced the performance of pronunciation in letter-to-phoneme conversion. However, automatic syllabification is difficult and still a challenging problem. Why is automatic syllabification from text hard? Various reasons are given in the research literature. Having identified the syllable boundary with the absence of precise definition of itself is one of the most cited reasons. The ambiguity whether syllabification from spelling or phonetic transcription is also an issue. The lack of a gold standard syllabified lexicon makes a syllabification algorithm difficult to learn from examples. Even, there is a dictionary to determine the syllabification by look-up technique, it is still a problem when encountering new words that do not have a corresponding entry in the dictionary. Further, the step for syllabification and pronunciation induction should be concerned with finding the pronunciation first followed by the syllabification, or vice versa, or finding both in parallel. These are the possible reasons that automatic syllabification is a very hard problem.

Many researchers have proposed a variety of data-driven approaches to assign syllable boundaries. In this chapter, an attempt was made to use the concept of analogy for syllabification and pronunciation of proper names. The series model of syllabification/pronunciation based on the PbA approach was evaluated with the set of proper names from CMUDICT. Using a leave-one-out strategy, the performance of PbA has not been improved relative to that of our standard model. The errors in automatically-inferred syllabification seem to disrupt the substring matching process in PbA. One possibility to improve the performance is to provide the manually-syllabified dictionary

Combination	00001	00010	00011	00100	00101	00110	00111	01000
00001	53.15	55.43	54.83	53.43	53.49	53.95	54.09	51.73
00010	54.22	55.64	55.36	54.38	54.67	54.71	55.01	51.69
00011	58.20	59.68	59.45	58.48	58.66	58.88	59.12	56.11
00100	59.66	60.43	59.80	60.21	59.94	60.01	60.11	57.44
00101	63.09	64.17	63.88	63.67	63.71	63.75	63.89	61.19
00110	58.63	59.52	59.41	59.17	59.49	59.34	59.62	56.34
00111	61.14	62.29	62.12	61.72	61.94	61.94	62.18	59.10
01000	36.46	37.87	37.35	36.45	36.08	36.92	36.74	36.56
01001	50.93	52.72	52.28	50.96	51.27	51.56	51.76	50.13
01010	54.58	55.79	55.70	54.74	55.17	55.32	55.58	52.31
01011	57.65	58.97	58.94	57.94	58.17	58.42	58.73	55.80
01100	58.62	59.35	59.17	59.13	59.18	59.28	59.40	56.72
01101	61.65	62.72	62.44	62.29	62.42	62.42	62.60	59.91
01110	58.36	59.26	59.22	58.91	59.30	59.15	59.43	56.09
01111	60.82	61.82	61.69	61.27	61.67	61.55	61.82	58.65
10000	57.60	59.35	58.92	57.99	58.11	58.41	58.54	56.25
10001	56.51	58.59	57.98	56.51	56.81	57.18	57.23	55.42
10010	59.04	60.42	60.38	59.37	59.79	59.82	60.04	57.16
10011	59.69	61.34	61.21	60.07	60.34	60.57	60.79	58.00
10100	63.63	64.52	64.35	64.21	64.37	64.25	64.38	61.96
10101	**63.88**	**65.14**	**64.93**	**64.47**	**64.60**	**64.73**	**64.83**	**62.17**
10110	61.36	62.42	62.33	61.78	62.16	62.11	62.27	59.53
10111	62.32	63.71	63.51	62.92	63.20	63.26	63.40	60.39
11000	54.75	56.51	56.02	54.76	55.14	55.42	55.58	54.49
11001	55.03	56.96	56.46	55.05	55.45	55.72	55.80	54.43
11010	58.82	60.20	60.07	59.12	59.51	59.58	59.85	57.18
11011	58.64	60.25	60.10	58.85	59.31	59.47	59.69	57.37
11100	63.38	64.22	64.02	63.79	64.10	63.93	64.09	61.62
11101	62.57	63.69	63.48	62.96	63.27	63.28	63.45	60.87
11110	61.59	62.68	62.58	62.00	62.36	62.30	62.41	59.66
11111	61.93	63.17	63.03	62.38	62.78	62.86	62.94	60.08

TABLE 4.1: Results of the series (S+P)bA model, in which pronunciation is inferred after syllabification by SbA. The columns represent the combinations of scoring strategy, ranging from 00001 to 01000, in which the results obtained from when inferring syllabification from Webster's dictionary. The rows represent the 31 combinations of scoring strategy when inferring pronunciation from CMUDICT proper name. The results are presented in terms of percentage word accuracy.

Combination	01001	01010	01011	01100	01101	01110	01111	10000
00001	53.14	55.14	55.06	53.41	53.78	54.00	54.51	53.33
00010	53.69	55.34	55.63	54.22	54.79	54.74	55.36	54.87
00011	57.78	59.43	59.56	58.55	58.69	58.95	59.20	58.78
00100	58.56	59.95	60.22	59.99	60.00	60.07	60.58	60.04
00101	62.48	63.97	64.02	63.62	63.70	63.77	64.05	63.70
00110	57.90	59.43	59.47	59.10	59.40	59.38	59.78	59.22
00111	60.63	62.19	62.19	61.75	61.92	61.97	62.21	61.83
01000	36.17	37.75	37.56	36.45	36.37	36.86	37.01	36.41
01001	51.04	52.72	52.47	51.09	51.58	51.68	51.93	51.05
01010	54.15	55.72	55.74	54.84	55.30	55.31	55.66	55.02
01011	57.31	58.95	59.05	58.04	58.28	58.48	58.68	58.11
01100	57.77	59.28	59.29	59.19	59.25	59.29	59.58	59.05
01101	61.23	62.63	62.70	62.24	62.39	62.48	62.66	62.33
01110	57.72	59.25	59.22	58.93	59.20	59.17	59.45	59.01
01111	60.33	61.78	61.85	61.33	61.64	61.60	61.80	61.38
10000	57.72	59.27	59.08	58.01	58.28	58.41	58.62	58.00
10001	56.57	58.39	58.21	56.71	57.06	57.23	57.47	56.57
10010	58.72	60.37	60.38	59.51	59.80	59.87	60.00	59.62
10011	59.36	61.28	61.25	60.22	60.39	60.64	60.74	60.32
10100	62.97	64.37	64.36	64.28	64.39	64.27	64.42	64.20
10101	**63.47**	**64.90**	**65.05**	**64.58**	**64.67**	**64.74**	**64.89**	**64.39**
10110	60.94	62.33	62.32	61.88	62.13	62.14	62.26	61.86
10111	61.89	63.65	63.59	63.10	63.16	63.27	63.37	63.03
11000	54.99	56.51	56.41	55.00	55.43	55.54	55.79	54.63
11001	55.15	56.98	56.75	55.25	55.65	55.73	56.03	54.93
11010	58.50	60.09	60.30	59.27	59.57	59.58	59.82	59.26
11011	58.49	60.29	60.25	59.17	59.41	59.54	59.69	59.14
11100	62.81	64.05	64.25	63.84	64.10	63.97	64.22	63.79
11101	62.21	63.55	63.72	63.04	63.32	63.27	63.53	62.99
11110	61.05	62.56	62.67	62.05	62.28	62.29	62.44	62.12
11111	61.52	63.07	63.09	62.60	62.76	62.83	62.90	62.55

TABLE 4.2: Results of the series (S+P)bA model, in which pronunciation is inferred after syllabification by SbA. The columns represent the combinations of scoring strategy, ranging from 01010 to 10000, in which the results obtained from when inferring syllabification from Webster's dictionary. The rows represent the 31 combinations of scoring strategy when inferring pronunciation from CMUDICT proper name. The results are presented in terms of percentage word accuracy.

Combination	10001	10010	10011	10100	10101	10110	10111	11000
00001	53.33	54.92	54.42	53.76	53.61	54.19	54.16	53.66
00010	54.60	55.83	55.50	54.99	54.58	55.30	55.47	54.97
00011	58.79	59.68	59.57	59.13	58.93	59.21	59.37	59.12
00100	59.76	60.41	59.96	60.43	60.03	60.48	60.56	59.63
00101	63.62	64.02	63.94	64.07	63.90	63.97	64.07	63.48
00110	59.14	59.76	59.65	59.66	59.38	59.75	59.87	59.23
00111	61.67	62.26	62.26	62.24	62.14	62.21	62.32	61.90
01000	36.03	37.58	37.06	36.47	36.13	37.03	36.81	36.43
01001	51.19	52.37	51.99	51.41	51.40	51.73	51.65	51.38
01010	54.98	56.05	55.85	55.26	55.15	55.66	55.75	55.35
01011	58.16	59.10	59.02	58.42	58.42	58.62	58.77	58.51
01100	58.91	59.44	59.20	59.46	59.29	59.56	59.63	59.00
01101	62.24	62.69	62.66	62.68	62.56	62.62	62.65	62.22
01110	58.86	59.56	59.39	59.37	59.19	59.54	59.59	59.05
01111	61.30	61.87	61.89	61.71	61.73	61.82	61.95	61.57
10000	57.92	59.08	58.79	58.22	58.28	58.60	58.54	58.47
10001	56.81	58.12	57.70	57.02	57.08	57.42	57.32	57.12
10010	59.68	60.42	60.44	59.98	59.87	60.07	60.20	59.87
10011	60.28	61.30	61.27	60.70	60.53	60.82	60.84	60.54
10100	64.15	64.50	64.44	64.43	64.45	64.41	64.46	64.27
10101	**64.54**	**65.08**	**65.04**	**64.78**	**64.91**	**64.90**	**64.93**	**64.52**
10110	61.93	62.40	62.46	62.23	62.21	62.35	62.43	62.16
10111	62.96	63.59	63.57	63.33	63.33	63.37	63.54	63.16
11000	55.04	56.12	55.76	55.21	55.16	55.60	55.56	55.52
11001	55.33	56.49	56.09	55.57	55.56	55.92	55.80	55.77
11010	59.50	60.19	60.14	59.68	59.56	59.92	60.04	59.64
11011	59.23	60.19	60.06	59.51	59.37	59.72	59.74	59.51
11100	63.90	64.25	64.19	64.24	64.11	64.13	64.25	64.01
11101	63.15	63.67	63.64	63.45	63.43	63.51	63.60	63.16
11110	62.30	62.63	62.69	62.42	62.42	62.50	62.61	62.23
11111	62.58	63.19	63.21	62.85	62.86	63.02	63.11	62.73

TABLE 4.3: Results of the series (S+P)bA model, in which pronunciation is inferred after syllabification by SbA. The columns represent the combinations of scoring strategy, ranging from 10001 to 11000, in which the results obtained from when inferring syllabification from Webster's dictionary. The rows represent the 31 combinations of scoring strategy when inferring pronunciation from CMUDICT proper name. The results are presented in terms of percentage word accuracy.

Combination	11001	11010	11011	11100	11101	11110	11111
00001	53.75	54.78	54.64	54.04	54.06	54.39	54.30
00010	54.65	55.63	55.98	55.16	55.31	55.50	55.62
00011	58.99	59.53	59.62	59.08	59.22	59.36	59.39
00100	59.71	60.09	60.40	60.50	60.43	60.54	60.63
00101	63.63	63.90	64.03	64.00	64.00	64.09	64.11
00110	59.11	59.64	59.80	59.72	59.69	59.86	60.00
00111	61.88	62.25	62.35	62.18	62.22	62.29	62.39
01000	36.28	37.43	37.32	36.60	36.51	37.06	37.01
01001	51.46	52.24	52.14	51.79	51.75	51.88	51.96
01010	55.01	55.92	56.02	55.48	55.52	55.76	55.87
01011	58.35	59.03	59.02	58.60	58.71	58.77	58.85
01100	58.90	59.37	59.40	59.61	59.40	59.67	59.74
01101	62.27	62.58	62.69	62.70	62.65	62.74	62.78
01110	58.88	59.42	59.49	59.48	59.39	59.61	59.66
01111	61.49	61.87	61.96	61.87	61.86	61.89	61.99
10000	58.27	59.08	58.82	58.35	58.40	58.72	58.65
10001	57.07	57.94	57.82	57.27	57.31	57.58	57.44
10010	59.69	60.45	60.54	59.90	59.96	60.23	60.27
10011	60.51	61.36	61.24	60.65	60.67	60.95	60.97
10100	64.11	64.41	64.49	64.46	64.46	64.48	64.57
10101	**64.49**	**65.03**	**65.10**	**64.93**	**64.85**	**64.01**	**64.03**
10110	62.00	62.44	62.50	62.25	62.30	62.44	62.53
10111	63.08	63.59	63.64	63.37	63.46	63.51	63.55
11000	55.39	56.13	56.00	55.43	55.47	55.80	55.79
11001	55.67	56.43	56.25	55.82	55.83	56.08	56.02
11010	59.52	60.13	60.16	59.68	59.74	60.09	60.06
11011	59.55	60.20	60.13	59.60	59.57	59.93	59.92
11100	63.87	64.23	64.36	64.27	64.22	64.33	64.36
11101	63.10	63.64	63.73	63.55	63.57	63.72	63.73
11110	62.22	62.61	62.71	62.52	62.54	62.65	62.69
11111	62.67	63.20	63.20	62.99	63.03	63.12	63.07

TABLE 4.4: Results of the series (S+P)bA model, in which pronunciation is inferred after syllabification by SbA. The columns represent the combinations of scoring strategy, ranging from 11001 to 11111, in which the results obtained from when inferring syllabification from Webster's dictionary. The rows represent the 31 combinations of scoring strategy when inferring pronunciation from CMUDICT proper name. The results are presented in terms of percentage word accuracy.

of proper name for inferring syllabification/pronunciation. Also, it is hoped that other methods for syllabification which can achieve an increased performance will be discovered in future work.

Chapter 5

Multilingual Pronunciation

5.1 Introduction

The difficulty of the automatic pronunciation problem is known to vary widely across different languages, according to the complexity of the relationship between pronunciation and orthography in each specific language. This is generally taken to vary across a so-called deep/shallow continuum (Coltheart 1978; Liberman et al. 1980; Katz and Feldman 1981; Turvey et al. 1984; Sampson 1985). A 'shallow' orthography means that the correspondences between letters and sounds (graphemes/phonemes) in the writing system are close to one-to-one (Davis 2005). For languages like English or French whose writing system is generally agreed to be 'deep', there is a supposedly complex relation between spelling and sound, unlike the 'shallow' orthographies of Finnish or Serbian, for example, where the correspondence is mostly if not entirely consistent and transparent. Thus, we expect that automatic pronunciation will be particularly difficult for English. However, it does seem to be relatively easier to convert spelling into sound for languages such as Spanish and Italian. So that the difficulty of letter-to-phoneme conversion varies from language to language. One potential advantage of data-driven method like PbA is that they are highly portable between different languages. All that is necessary is to change the lexicon that acts as the source of example pronunciations. To date, the success of PbA for multilingual pronunciation generation has not been seriously assessed. Hence, one goal for this thesis is to study PbA performance on multilingual transcription as a way of quantifying the variation of difficulty of the task across languages, and gaining insight into manifestations of the deep/shallow continuum.

Since PbA uses a dictionary of example spellings and pronunciations as its knowledge base, an important question is what size of dictionary we should employ in a text-to-speech system. Intuitively, we might feel that the larger it is, the better. However, large dictionaries are expensive to compile, lead to an increase in processing time, and may not exist for all languages that we wish to synthesise, especially minority languages. Also,

there are inherent dangers in extrapolating from results on a small database or dictionary to asymptotic performance on a very large dictionary. A few years ago, Baayen (2001, p.xxi) wrote:

> 'Word frequency distributions are characterised by very large numbers of rare words. This property leads to strange phenomena such as mean frequencies that systematically change as the number of observations is increased, relative frequencies that even in large samples are not fully reliable estimators of population probabilities, and model parameters that vary with text or corpus size'.

The problems that this phenomenon can cause for speech synthesis have often gone unrecognised or underestimated (Möbius 2003). For instance, early developers of rule-based letter-to-sound systems tested on small datasets and assumed that error rates would be independent of test set size, leading to dramatic over-estimates of performance (Damper et al. 1999). With the increased interest in data-driven approaches (Damper 2001, p.xiii), an important issue becomes the sizes of the training and test sets if, as Baayen (2001, p.xxi) says 'model parameters ... vary with ... corpus size'. So although it is likely, and some preliminary results from Damper et al. (1999) suggest it is the case, it is by no means certain that 'bigger is better'.

Given this background, our purposes in this chapter are two-fold:

1. To evaluate PbA on a range of different languages, so as to quantify the variation of transcription difficulty across the deep/shallow continuum of orthography.
2. Also, to explore the effect of lexicon size on performance for multilingual transcription using PbA.

Specifically, we have investigated the performance of PbA applied to 7 European languages, which are Dutch, English, French, Frisian, German, Norwegian, and Spanish—with 12 different lexicons. Also, we artificially varied the size of (some of) these lexicons by evaluating transcription accuracy on different subsets of the complete dictionary. Ideally, we aim to investigate performance on the lexicons of proper names in each languages. However, such dictionaries are not available in every language. In this work, dictionaries of common words were used instead of proper names except one for English language source (CMUDICT), which includes both types of word. In the next section, the dictionaries used in this work are described. In Section 5.3, we set out the various evaluations of transcription accuracy that have been performed. The results are presented and discussed in Section 5.4. Conclusions and suggestions are presented in Section 5.5.

Language / Lexicon		Number of ...		
		Letters	Phonemes	Words
	Dutch	43	44	116,252
	Frisian	39	85	61,976
	German	31	59	49,421
	Norwegian	29	47	41,713
	Spanish	33	26	31,491
French:	Lexique	40	39	36,460
	Brulex	40	39	27,473
	Novlex	38	40	9,447
English:	BEEP	26	43	198,632
	CMUDICT	26	39	112,091
	Webster	26	51	20,008
	TWB	26	51	16,280

TABLE 5.1: Number of letters, phonemes and word types in each dictionary.

5.2 Lexical Databases

The lexicons used in this research vary in size from about 9,000 to almost 200,000 words, and are all available at `http://www.pascal-network.org/Challenges/PRONALSYL/Datasets/`. We have used the automatically-aligned dictionaries for seven languages: Dutch, English, French, Frisian, German, Norwegian, and Spanish. In total, twelve different dictionaries are used in this work. For French, we have used Lexique, Brulex, and Novlex. For English, we have used the British English Example Pronunciation (BEEP) dictionary, CMUDICT from Carnegie-Mellon University, Webster's, and Teachers' Word Book (TWB). The phoneme sets used for these lexicons are different, even for the same language. For the other five languages, there is only one lexicon per language. The letters and phonemes in all dictionaries are automatically aligned using the algorithm of Damper et al. (2005), except in the case of Webster, used in NETtalk, and TWB, used in NETspeak, which were manually aligned by the original authors. Since most of these dictionaries do not include stress and/or syllable boundary markers, these aspects of the transcription task have had to be ignored, in spite of their obvious importance.

Table 5.2 summarises the number of letter, phoneme and word types in each dictionary. It can be seen that there is wide variation in the phoneme inventory, not only between languages but also between different dictionaries for the same language.

5.3 Experimental Design

In line with the two goals previously stated, the following evaluations were conducted:

1. Transcription performance was evaluated on the complete dictionary for each of

the twelve dictionaries covering the seven languages. The purpose here was to quantify the variation of transcription difficulty across the deep/shallow continuum of orthography represented by these seven languages.

2. For each of the seven languages, transcription performance was evaluated as a function of dictionary size. The purpose here was to explore the effect of lexicon size on performance for multilingual transcription using PbA.

Regarding 2, where there are multiple dictionaries for a language (i.e., French and English), the largest available dictionary (i.e., Lexique and BEEP, respectively) is selected. Dictionary size was then varied artificially by randomly dividing the dictionary for language l into 10 approximately equal size partitions, or 'folds', $\mathcal{P}_1^l, \mathcal{P}_2^l, \ldots \mathcal{P}_{10}^l$. Ten different-sized subsets were then formed as $\mathcal{P}_1^l, (\mathcal{P}_1^l \cup \mathcal{P}_2^l), \ldots, (\mathcal{P}_1^l \cup \mathcal{P}_2^l \cup \ldots \cup \mathcal{P}_{10}^l)$. Because the size of each of the seven dictionaries is not the same, it follows that, in general, $|\mathcal{P}_1^m| \neq |\mathcal{P}_1^n|$, $m \neq n$. Because the dictionary sizes for each language are not necessarily exactly divisible by ten, in general, the tenth partition for a language is smaller in size than the other nine partitions:

$$|\mathcal{P}_{10}^l| \leq |\mathcal{P}_9^l| = |\mathcal{P}_8^l| = \cdots = |\mathcal{P}_1^l|$$

5.4 Experimental Results

In this section, we present the results of applying PbA to the lexicons described in the previous section. These are reported in terms of words and phonemes correct. Transcription accuracy was evaluated using both a leave-one-out strategy and 10-fold cross validation (Cherkassky and Mulier 1998). That is, in the case of a leave-one-out method, each word was removed in turn from the dictionary and a pronunciation derived from the remaining words. In the case of 10-fold cross validation, each of the 12 dictionaries was divided into 10 partitions (folds), as described in the previous section. Each fold was removed in turn and used as a test set; the remaining nine folds acting as the dictionary for inferring pronunciations. It should be obvious that leave-one-out is also a form of k-fold cross-validation where k is equal to the number of entries in the dictionary.

5.4.1 Results on the 12 different dictionaries

Table 5.1(a) summarises results on all 12 dictionaries using the leave-one-out method. The corresponding results obtained by averaging across the 10 folds shown in Table 5.1(b) were very similar (a fraction of one percentage point in all but one case), although they are consistently lower. The binary coding in the final column of these tables indicates

(a) Leave-one-out

Language / Lexicon		% accuracy		Best combination
		Word	Phoneme	
	Dutch	94.43	99.20	11111
	Frisian	85.18	97.58	10101
	German	92.94	98.94	10101
	Norwegian	95.05	99.09	10100
	Spanish	99.43	99.80	11011/11101
French	Lexique	91.31	98.18	11100
	Brulex	91.95	98.34	10101
	Novlex	86.94	96.47	11100
English:	BEEP	87.50	98.43	10100
	CMUDICT	72.13	95.56	11111
	Webster's	65.46	92.42	11111
	TWB	71.98	94.36	11100

(b) 10-fold cross validation

Language / Lexicon		% accuracy		Best combination
		Word	Phoneme	
	Dutch	94.34 (0.184)	99.18 (0.033)	11111
	Frisian	84.60 (0.434)	97.47 (0.080)	10101
	German	92.74 (0.464)	98.91 (0.070)	10101
	Norwegian	94.94 (0.243)	99.05 (0.068)	10100
	Spanish	99.38 (0.161/0.161)	99.78 (0.090/0.090)	11011/11101
French:	Lexique	91.02 (0.339/0.399)	98.10 (0.076/0.075)	11100/11101
	Brulex	91.78 (0.487)	98.29 (0.130)	10101
	Novlex	86.53 (1.666)	96.34 (0.747)	11100
English:	BEEP	87.31 (0.257)	98.41 (0.034)	10100
	CMUDICT	71.99 (0.485)	95.53 (0.110)	10101
	Webster's	64.52 (1.512)	92.16 (0.325)	10111
	TWB	70.77 (1.121)	94.08 (0.332)	11100

TABLE 5.2: Results of applying PbA to 12 dictionaries. Accuracies for 10-fold cross-validation in (b) are averages across the 10 folds with standard deviations in brackets.

which combination of PbA heuristic scoring strategies gave best performance. A 1 in position p of the binary code indicates that the pth strategy in Marchand and Damper (2000) was included in the rank-fusion combination; a 0 indicates that it was not.

Broadly in line with expectations based on our initial intuitions about the relative difficulty of letter-to-phoneme conversion in different languages, the best results are achieved for Spanish at $> 99\%$ word accuracy and the lowest performance is obtained for English. Performance for the other languages (Dutch, French, German, Norwegian) was generally at $> 90\%$ words correct, whereas for Frisian the result was $\sim 85\%$ words correct. There are few data in the literature on the problem of multilingual letter-to-phoneme conversion with which to compare our results. One exception is van den Bosch et al. (1994) who attempt to measure the complexity of the French, Dutch and English writing systems based on the two measures of success at letter-phoneme alignment and

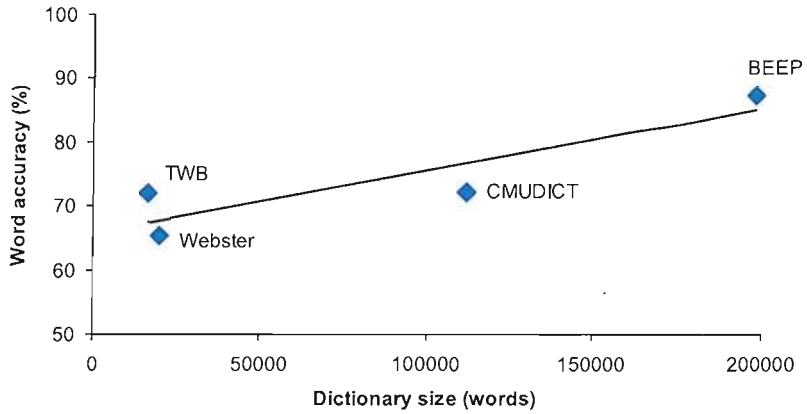

FIGURE 5.1: Variation of word accuracy with dictionary size for English letter-to-phoneme transcription, showing best fit regression line.

accuracy of letter-to-phoneme conversion. Generally, they find that French is easier to transcribe from spelling to pronunciation than Dutch which in turn is easier than English. Our results are somewhat different in that the relative difficulty of French and Dutch are reversed in our data. Obviously, some differences are to be expected in light of the use of different dictionaries and different methods for automatic transcription.

For French and English, the results vary across the dictionaries; the variation is especially wide for English. Factors accounting for this variation are likely to include:

- the different sizes of the dictionaries;
- the different sizes of the phoneme inventories;
- differing transcription standards employed by the dictionary compilers.

Figure 5.1 shows the variation of word accuracy with dictionary size for English. There is a reasonably strong positive correlation ($R^2 = 0.797$) between accuracy and size, showing that this factor seems to have a real effect. We speculate that larger dictionaries have lower complexity in that the extra words are likely to be morphologically related to other entries, and this lower complexity is reflected in higher transcription accuracy.

Figure 5.2 shows the variation of word accuracy with the size of phoneme inventory employed by the dictionary, again for English. Here, there is a relatively much weaker negative correlation ($R^2 = 0.225$) between accuracy and size of the phoneme set. Some such negative correlation is only to be expected; the lower the size of the phoneme inventory, the broader is the transcription standard being used, and so the less potential there is for phoneme substitution errors.

We have not attempted any similar analysis of the results for French because of the fewer number of dictionaries employed (three rather than four, with two being very similar

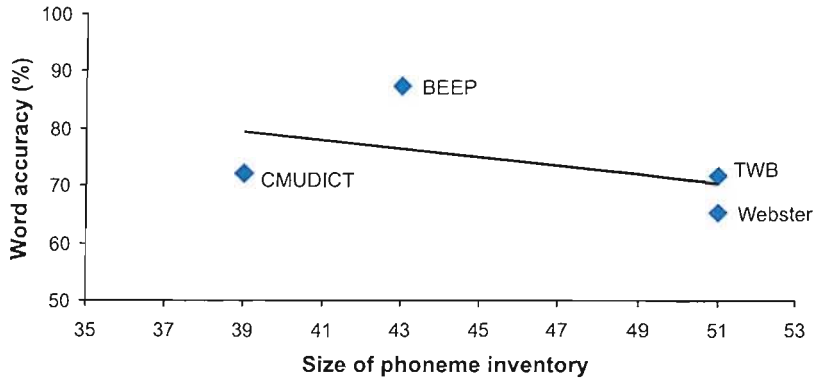

FIGURE 5.2: Variation of word accuracy with size of dictionary phoneme inventory for English letter-to-phoneme transcription, showing best fit regression line.

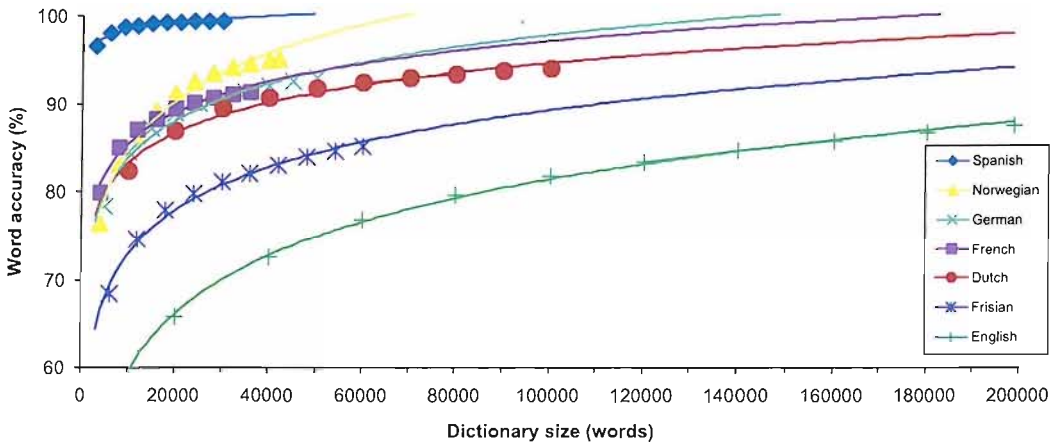

FIGURE 5.3: Variation of word accuracy with different sizes of dictionaries in seven languages.

in size), and because the variation in transcription performance and in size of phoneme inventory is much less than for English.

5.4.2 Results as a function of dictionary size

Figure 5.3 shows the variation of word transcription accuracy in the seven languages as a function of dictionary size, with different-size dictionaries constructed as described in Section 5.3. The results shown here were obtained using leave-one-out and the best combination as tabulated in Table 5.1(a).

There is a clear and obvious tendency for transcription performance to grow monotonically with dictionary size. This goes some way to explaining why the 10-fold cross validation results in Table 5.1(b) are consistently very slightly smaller than the leave-

Language	α	β	R^2
Spanish	1.12	88.1	0.9184
Norwegian	7.99	11.2	0.9850
German	6.10	27.6	0.9752
French	4.97	39.7	0.9759
Dutch	4.81	39.0	0.9746
Frisian	7.02	8.22	0.9893
English	9.52	−28.23	0.9984

TABLE 5.3: Best-fit parameters for the regression model $T = \alpha \ln S + \beta$.

one-out results in Table 5.1(a), because the size of dictionary used for inference is smaller than with leave-one-out. It is $\frac{9}{10}$ths the size of the complete dictionary.

For each language, the data are well-modelled by a function of the form:

$$T = \alpha \ln S + \beta \tag{5.1}$$

where T is percentage transcription accuracy, S is lexicon size, and α and β are language-dependent regression parameters, tabulated in Table 5.4.2. As can be seen from the final column of the table, the fit to the mathematical model of equation (5.1) is excellent, with $R^2 > 0.9$ in all cases.

In spite of the high R^2 correlation coefficients obtained, there is one obvious sense in which this model is deficient: The logarithmic function does not saturate (although it does decelerate) as S increases, whereas the actual transcription accuracy obtained cannot exceed 100%. This deficiency in the model is clearly seen in the curve for Norwegian in Fig. 5.3, where the extrapolated best-fit curve appears to be tending to a value well above 100%. The situation bears similarities to the mathematical modelling of lexicon coverage in the earlier work of Damper et al. (1999). In this case, consideration of Zipf's law led to a logarithmic model like (5.1) that was limited by setting a parameter (effectively α) according to the total number of words in the language. This concept of "the total number of words in the language" is, of course, problematic from a theoretical point of view. The unbounded set of all words in any language makes it impossible to list every word. In the present situation, the growth of transcription accuracy T can be similarly limited by appropriate setting of α and β, assuming that S can never exceed some upper bound. Although this artificial device is rather unsatisfactory, it is interesting that very similar problems arise in the two cases.

In equation (5.1), α controls the rate of growth of transcription accuracy whereas β controls the vertical placement of the growth curve. From this perspective, we would expect a language possessing shallow orthography to display a high value of β, probably in conjunction with a low value of α (since high transcription accuracy will already be

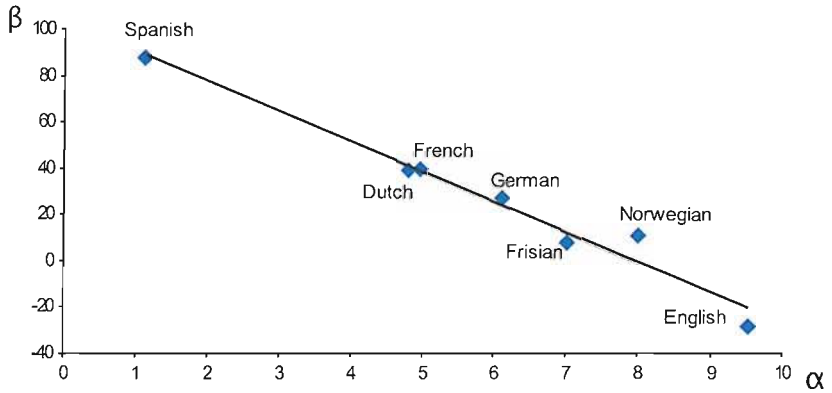

FIGURE 5.4: Relation between α and β for seven European languages.

achieved for a relatively small lexicon). This is precisely the pattern seen for Spanish (Table 5.4.2). On the other hand, a language with deep orthography should show a low value of β; it is less clear how this would couple with the value of α. One might expect that α would be low (i.e., low growth) because transcription is 'difficult'; alternatively, one might predict that α would be relatively large because growth is from a lower value for $S \sim$ a few thousand words.

To explore this issue, we plot β versus α in Figure 5.4, whereupon a clear linear trend between the two parameters of the form is found:

$$\beta = -13.069\alpha + 104.05 \tag{5.2}$$

with $R^2 > 0.9701$. There was no particular reason that we can see to expect any such relation *a priori*, since (as outlined above) we interpreted one parameter to control growth rate and the other to control vertical placement. With hindsight, however, it makes sense for there to be a dependence between the two, with larger growth to the 100% asymptote for a language with a deep orthography, starting from a relatively low transcription accuracy for a small dictionary.

Substituting 5.2 into 5.1 to eliminate β gives the model:

$$T \sim 100 - \alpha\left(13 + \ln\frac{1}{S}\right) \tag{5.3}$$

This is an interesting form, indicating that transcription accuracy is limited to 100% for $S < e^{13}$ (approximately 450,000) independent of the value α.

Note that (5.3) means that this is a one-parameter model to fit only the data of Figure 5.3. Assuming the constant of $\sim$ 100 is playing the role of setting asymptotic performance, the constant of 13 in (5.3) should be viewed as another 'parameter' which

Language	α	$Rank_{\alpha}$	$Rank_{asymp}$	$Rank_{low}$
Spanish	1.12	1	1	1
Norwegian	7.99	6	2	3.5
German	6.10	4	3.5	3.5
French	4.97	3	3.5	3.5
Dutch	4.81	2	5	3.5
Frisian	7.02	5	6	6
English	9.52	7	7	7

TABLE 5.4: Ranking of α values, asymptotic transcription accuracy and 'low' transcription accuracy on a small dictionary.

turns out empirically to be the same across all seven languages studied here. Further work is required to determine how general this is across a wider range of languages.

The question then arises: How good a measure is the language-dependent parameter α of the depth of orthography (or the difficulty of letter-to-phoneme transcription) for that language? Table 5.4 shows the seven languages and ranks assigned to the value of α obtained by regression ($Rank_{\alpha}$) and to the difficulty of transcription ($Rank_{asymp}$) assigned according to the ordering of asymptotic performance in Figure 5.3. For α, ranking is in ascending order; for transcription accuracy, it is in descending order. Note that we have ranked Spanish as easier to transcribe than Norwegian (in spite of a slower rate of deceleration of its T-S curve) as the regression for the latter language looks suspect, and we have considered French and German to tie as it is difficult to separate the performance for these two languages. Let the null hypothesis be that there is no relation between the asymptotic difficult of transcription and α. By the Spearman rank correlation test (Siegel and Castellan 1988, p.202-213), $r_s = 0.5335$ is obtained according to which there is no reason to reject the null hypothesis.

However, the asymptotic performance in Figure 5.3 is possibly unreliable, being based on extrapolation from a model fitted to empirical data. An alternative, preferable measure of the degree of transcription difficulty might be the 'low' measure obtained on a dictionary of about 10,000 words, $Rank_{low}$, where at least we have actual data. Since it turns out to be rather difficult to separate Norwegian, French, German and Dutch at $S = 10,000$ in Figure 5.3, these have been treated as tied on $Rank_{low}$. This yields $r_s = 0.8078$, allowing us to reject the null hypothesis at the 5% level of significance. Hence, α for a language appears to be a good predictor of performance on a small dictionary of that language.

5.5 Conclusions

In this chapter, we have studied the variation of accuracy of transcription across 12 lexicons from 7 European languages (Dutch, English, French, Frisian, German, Norwegian,

and Spanish). The success of the transcription task is assumed to reflect the complexity of the writing system (i.e., the deep/shallow continuum of orthography) of the particular language. Also, the size of learning data set seems to be the main factor impacting results. Thus, we have studied the full extent of the relationship between size and accuracy. The selected lexicons from these languages are divided into 10 different 'sizes' of dictionaries and evaluated. These results give an idea of the reasonable size of lexicon that should be used to compromise a trade-off between the performance and the processing time of PbA approach.

Considering the effect of lexicon size on performance for seven languages, the size of phoneme set used by the dictionary compilers can have an effect, as shown in the results of Section 5.4.1. However, when different-sized lexicons are constructed as unions of 10 folds of the same dictionary (Section 5.3), the results in Section 5.4.2 show that transcription accuracy increases monotonically with the size of the lexicon used for analogical inferencing. These results also suggest that the bigger the size of dictionary, the better the performance achieved.

Although this simple result might be thought unsurprising, there are good reasons for treating it as something other than vacuous. The LNRE phenomenon (Baayen 2001, p.51-57; Möbius 2003) means that simple-minded assumptions about how parameters of a language model grow with corpus size are dangerous. Further, it is one thing to assume a relationship and quite another to demonstrate that it holds empirically. Finally, test and training dataset sizes can have a profound effect on results of data-driven approaches to language learning. This is most obviously the case in eager learning methodologies, like neural networks, where overfitting to the training data is an ever-present danger. It seem that yet another advantage of lazy learning is the avoidance of the over-regularisation with can result from a prior training phase (Daelemans et al. 1999).

With reference to the variation of transcription difficulty across these seven languages, this work may be one of only very few attempts to quantify depth of orthography computationally. In line with general beliefs in the field, it is found that Spanish is at one extreme of the deep/shallow continuum for the languages tested, whereas English is at the other. English is notorious for the lack of regularity in its spelling-to-sound correspondence, which largely reflects the many complex historical influences on the spelling system (Venezky 1965, p.3-4; Scragg 1975, p.12; Carney 1994, p.1,p.12).. Indeed, Abercrombie (1981) describes English orthography as "one of the least successful applications of the Roman alphabet." This is reflected in a very large value for the language-dependent parameter α in equation (5.3) of 9.52, where as for Spanish it is $\alpha = 1.12$. The case of Norwegian appears somewhat anomalous, having a high value for α (7.99, the second highest found in this work) but seeming to be about as easy to transcribing as French, German and Dutch according to the results displayed in Figure 5.3. This remarkable raises a question that an apparently anomalous value of

α is a genuine feature of the language or it is an artefact of some idiosyncrasy of the particular dictionary used here. Further work is needed on this point. It is also noticeable that its rate of deceleration towards asymptotic performance seems to slow.

Although (the case of Norwegian notwithstanding) α seems to be a reasonably good predictor of transcription performance on a dictionary of $\sim$ 10,000 words, it is less good at quantifying the asymptotic performance. This may be because the measure of asymptotic performance used here is unreliable, being based on extrapolation from the fitted regression model, quite distant from supporting data points. This poorly-fitting line probably suggests that a single function may not apply well to all languages. It is also possible that some other function might fit for all languages, instead of the one that we used here. A better extrapolated function is an important aspect that should be further investigated.

The PbA method is inspired by a model of reading aloud, suggesting that unknown words might be pronounced by analogy to real words that they resemble. For instance, adults who know the pronunciations of many words tend to pronounce an unseen word more correctly than children who know only a few. This hypothesis is affirmed by the results on different sizes of dictionaries. Considering the number of lexicon entries, bigger size would yield better performance of PbA. Not only the lexicon size affects the performance of PbA, but also the deep/shallow continuum plays as a key role in the success of PbA. Reading text in a language with shallow orthography would be easier for non-native speaker than reading text in a language with deep orthography. As Davis (2005) wrote, 'Finnish provides a good example, with 23 associations that match the exact number of letters. This effectively means that a non-Finnish person, who is a fluent reader in his/her own language, would be capable of reading aloud a Finnish text and make it perfectly comprehensible to a Finnish listener.' This assumption is reflected in the effectiveness of PbA across 7 languages. It is noticeable that the highest performance is obtained for Spanish, a language well known to have a shallow orthography. Additionally, the poorest result is achieved for English, which is notorious as a language with a deep orthography.

Chapter 6

Effect of Lexicon Composition in PbA

6.1 Introduction

So far, many variants of PbA have been proposed and evaluated with different lexicons. In practice, when encountering an unknown word in the system input, it is unlikely to know if it is a proper name or a technical word, or a common word. It should be possible to develop techniques for automatic classification, but these will never be entirely error-free. Therefore, one of several aspects to investigating the performance of PbA is to determine whether or not it makes a difference when the system infers a pronunciation by analogy with a lexicon containing:

1. known common words only,
2. known proper names only, or
3. a mix of common words and proper names.

If high accuracy can be obtained in case 3, then automatic classification of unknown words (with attendant potential for errors) might be avoided. Since PbA infers pronunciations using lexical words most similar (in an analogical sense) to the unknown word, there is a reasonable chance of this. In this chapter, we test this possibility, focusing on the effect that lexicon composition has on pronunciation accuracy for PbA.

6.2 Lexical Databases

Two publically-available dictionaries of pronunciations have been used in this work: BEEP containing common words and the CMU dictionary containing common words

and proper names. The description of these two dictionaries was previously detailed in Chapter 5. However, information about these dictionaries is repeated again and some more details are added here for the sake of clarity. These two dictionaries are used because BEEP is intended to document British English pronunciations, whereas CMUDICT contains American English pronunciations. We have also studied proper-name and common-word subsets of CMUDICT and mixtures of BEEP and CMU proper-name subset.

6.2.1 BEEP

BEEP is originally available as file `beep.tar.gz` from `ftp://svr-ftp.eng.cam.ac.uk/comp.speech/dictionaries/`. It contains approximately 250,000 word spellings and their transcriptions. After removing some words that contain non-letter symbols and/or words with multiple pronunciations, the number of words used in this work is 198,632. The phoneme set for BEEP consists of 44 symbols.

6.2.2 CMUDICT

CMU dictionary contains both common words and proper names, and their phonemic transcriptions. The phoneme set for CMU contains 39 symbols. The latest version (CMU version 0.6) can be downloaded from `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`. There are some duplicate words, some containing non-letter symbols and some where the pronunciation obviously does not match the spelling. These were removed to leave 112,091 words. In this chapter, CMUDICT is partitioned into two subsets as follows.

Proper Name Subset : there is no single, easily-available list of proper names and their pronunciations. However, a proper-name dictionary can be developed by using a list of proper names (without pronunciations) together with the standard CMU version 0.6. The list of names can be downloaded as file `cmunames.lex.gz` from `http://www.festvox.org`. It includes the most frequent names and surnames in the USA and their pronunciations (Font Llitjós 2001), from a wide variety of origins. The procedure was simply to extract from CMU pronunciations for the names on the first list. (Note, however, that some names on this list were not found in CMU.) This subset of CMUDICT is referred as Names. The number of proper names in Names is 52,911.

Common Word Subset : after extracting the proper names from CMU as above, the remaining words form the common word subset of 59,180 words. This dictionary is called Com.

BEEP	CMU	IPA	BEEP	CMU	IPA
aa	AA	ɑ	k	K	k
ae	AE	a	l	L	l
ah	AH	ʌ	m	M	m
ao	AO	ɔ	n	N	n
aw	AW	aʊ	ng	NG	ŋ
ax	**ER**	**ə**	**oh**	**AA**	**ɒ**
ay	AY	aɪ	ow	OW	oʊ
b	B	b	oy	OY	ɒɪ
ch	CH	tʃ	p	P	p
d	D	d	r	R	r
dh	DH	ð	s	S	s
ea	**EH**	**ɛ**	sh	SH	ʃ
eh	EH	ɛ	t	T	t
er	ER	ɜ	th	TH	θ
ey	EY	eɪ	**ua**	**AO**	**ʊ**
f	F	f	uh	UH	ʊ
g	G	g	uw	UW	u
hh	HH	h	v	V	v
ia	**IH**	**ɪ**	w	W	w
ih	IH	ɪ	y	Y	j
iy	IY	i	z	Z	z
jh	JH	ʤ	zh	ZH	ʒ

TABLE 6.1: Harmonisation scheme used to map the BEEP phoneme set onto the CMUDICT set.

6.2.3 Mixture Dictionary

This dictionary is a combination of BEEP and Names dictionaries as mentioned above. Because of the different phoneme sets between these two dictionaries, we need to collapse the BEEP phoneme set into the smaller of the two sets, which is the CMU phoneme set as specified in Table 6.1. This process is called harmonisation (Damper et al. 1999). As can be seen, five phoneme symbols of the BEEP phoneme set are collapsed into the CMU phoneme symbols that have the similar sounds, which are shown in bold. This dictionary is referred as Mixture.

6.3 Experimental Results

Performance was evaluated using a leave-one-out strategy and results are reported in terms of words correct. Stress assignment has been ignored for simplicity.

Table 6.2 shows the results of PbA with BEEP, Names and the Mixture dictionary in the form of a confusion matrix for all combinations of the three dictionaries as test set and lexical database. It should be noted that all entries are significantly different from

	Lexicon		
Test set	BEEP	Names	Mixture
BEEP	87.50	15.93	83.62
Names	23.57	68.35	55.08
Mixture	73.34	26.62	78.08

TABLE 6.2: Percentage words correctly transcribed by PbA with BEEP, Names and Mixture dictionaries.

one another (binomial tests, one-tailed, $p << 0.01$). As can be seen, best results for a given test-set dictionary are achieved when the same dictionary is used as the lexical database. Much higher accuracy is achieved when BEEP is used as the test set and lexical database (87.50% words correct) than when Names is used as the test set and lexical database (68.35% words correct). This is to be expected in view of the diversity of origin of the proper names and different degrees of assimilation into English (Vitale 1991; Spiegel 2003), making their pronunciation harder to infer. Cross-lexicon test/inference leads to a very large deterioration in performance. Although it is tempting to think that this indicates that proper names transcription is a harder problem than common word transcription, the difference could be due primarily or solely to the different sizes of lexicon, since PbA transcription accuracy is a strong function of dictionary size, increasing as the size of dictionary increases (see the previous chapter).

Using the Mixtures dictionary as test set and lexical database reflects the practical situation in which no attempt is made to classify the word class, merely treating all words as from the same class. Here the relevant result is 78.08% words correct, a long way below the performance when words from BEEP are pronounced by analogy with the entire BEEP dictionary. Note that a simple weighted linear sum of the BEEP/BEEP and Names/Names results (where the weights are the proportions of the two classes of word) would predict a result of 83.50% words correct, some way above the 78.08% result actually obtained. In effect, this weighted linear sum forms an upper bound on the performance that could be obtained if we had a perfect means of identifying the class of any input word.

In the results of the previous paragraph, the Mixture dictionary is of course heterogeneous, consisting of a British English lexicon of common words (whose phoneme set has had to be harmonised to CMU) and an American English dictionary of proper names. This was done to have the largest possible dictionaries. The performance of PbA have also been studied when the three dictionaries (common words, proper names and mixture) are homogeneous, all being derived from CMU. That is, Com, Names and CMU were used as the three dictionaries. Table 6.3 shows the corresponding results.

Here, the pattern of the results is the same as those of PbA with BEEP, Names and the Mixture dictionary. Again, it should be noted that all entries are also significantly different from one another (binomial tests, one-tailed, $p << 0.01$). The highest accuracy

	Lexicon		
Test set	Com	Names	CMU
Com	75.67	28.20	73.72
Names	38.63	68.35	64.96
CMU	58.12	47.15	69.61

TABLE 6.3: Percentage words correctly transcribed by PbA with Com, Names and CMU dictionaries.

is achieved when testing Com words against the Com lexicon itself. This Com vs. Com result (75.67% words correct) is much higher than Name vs. Names (68.35% words correct). This firmly supports the assumption that proper names are more difficult to pronounce than common words. Since the results are in the same way as those of heterogeneous sets, and the number of words in COM and Name are quite similar. Cross-lexicon test yields an expected low result in percentage of words correct. These results are slightly higher than in those of the cross-lexicon test with BEEP and Names. When testing COM or Names against the full CMU dictionary, deterioration is smaller than the corresponding case of the Mixture dictionary. We inclined to believe that the difference is due to the inhomogeneity of the latter (Mixture) dictionary, and the avoidance of harmonisation for Com/CMU. A huge drop in performance when testing Names against CMU firmly indicates that proper names have some special characteristics different from common words, as expected from their diversity.

Turning finally to the result of most practical interest, that is the third case mentioned in Section 6.1, using the full CMU dictionary as the lexical database. This reflects the situation where we have a single, undivided lexicon in the TTS system. Here, the relevant figures are 69.61% words correct when testing with CMU itself. This figure is lower than the result of 72.12% words correct that we would predict from a weighted linear sum of the Com vs. Com and Names vs. Names results. The Com/CMU result reflects the situation when input words are common words, we can get 73.72% words correct, but if the right automatic inference of input-word class is made, we can get 75.67% words correct instead. To the same extent, there is also a great loss in accuracy when testing Names against the full CMU dictionary, if no attempt is made to classify the class of input word. That is, testing Names against the CMU lexicon gave 64.96% words correct; testing Names against itself gave 68.35% words correct.

With reference to the performance on a mix dictionary of common words and proper names, in both heterogeneous and homogeneous, it suggests that automatic inference of input-word class would be advantageous for accuracy of PbA in TTS synthesis. However, we need to be aware of the dangers of misclassification, since this would lead to a very large deterioration in performance.

6.4 Conclusion

Pronunciation by analogy has been tested with different lexicon compositions: common words only, proper names only, and a mixture of the two. Two different dictionaries have been used: the large BEEP dictionary containing common words with their pronunciations for British English, and the CMU dictionary containing common words and proper names with their pronunciations for American English. BEEP was exploited because of the existence of the large dictionary; the attempt was complicated by the absence of a list of proper names. Thus, we believe that more credible results are given to those for the CMU dictionary. However, the results of both dictionaries are very similar. Excellent performance has been obtained when testing a dictionary of common words against itself or a dictionary of proper names against itself. Treating all words as from the same class, that is the mixed lexicon is used in all case, leads to a noticeable deterioration in performance. These experimental results recommend that automatic word class categorisation (common word vs. proper name) is useful for PbA approach. This could be beneficial for any analogy-based approach in the phonetic transcription module as well.

These studies also provide empirical support for the assumption that pronunciation of proper names is much more difficult than common words. Many reasons behind this assumption are given by many researchers. For instance, the most commonly cited reason is that for a specific language (e.g., American English) names originate from very different language families (Vitale 1991). The pronunciations of those names may assimilate to the phonological system of the new language over the years, thus names are often pronounced differently from their origins of languages (Font Llitjós 2001). Furthermore, there is more than one correct pronunciation for any given name depending on personal preference or regional influences (Spiegel 2003). Name pronunciation is also known to be idiosyncratic, that is many pronunciations contradict the phonological patterns of common words (Font Llitjós 2001). Particularly, the last presumption is affirmed by the results in this experiment when testing proper names alone against common words only or proper names against a combination of common-word and proper-name.

Chapter 7

Objective Evaluation

7.1 Introduction

In this chapter, PbA is compared with three other data-driven methods for proper name pronunciation, namely: the decision tree method described by Black et al. (1998), the table look-up method by Weijters (1991), and the table look-up method by van den Bosch and Daelemans (1993). These methods were selected on the basis that, first, we believe PbA, our main study approach, to be the best currently-available technique for pronunciation of common words (Damper et al. 1999) and, second, table look-up and decision trees are very representative of the competitor data-driven techniques to PbA. Indeed, table look-up can be seen as an alternative implementation of the broad concept of 'analogy', and also decision trees can be interpreted as a framework for building letter-to-sound rules. The comparison involves both objective and subjective performance. The objective results are reported and discussed here. The subjective evaluation is detailed in the next chapter.

For the PbA approach, the algorithm was exhaustively described in Chapter 3. The other three automatic pronunciation approaches that were used for objective and subjective evaluation are briefly described in the next section. Section 7.3 describes the re-implementation details of each method. Then, we briefly describe how the proper name dictionary of manually-supplied pronunciations was constructed. The results of objective evaluation are presented in Section 7.5. The conclusion is in Section 7.6.

7.2 Overview of the Techniques

In this section, we describe the three automatic pronunciation approaches. Data-driven approaches to letter-to-phoneme conversion generally require the letters of each word in the dictionary to be aligned with the corresponding phonemes, so converting the problem

Phoneme	7-gram vector						
-	-	-	-	**A**	A	R	D
AA	-	-	A	**A**	R	D	E
R	-	A	R	**D**	E	M	A
D	A	A	R	**D**	E	M	A
EH	A	R	D	**E**	M	A	-
M	R	D	E	**M**	A	-	-
AH	D	E	M	**A**	-	-	-

FIGURE 7.1: Complete set of 7-gram learning vectors for the name AARDEMA, pronounced /AA R D EH M AH/. The first element of the vector is the phoneme corresponding to the 4th letter in the 7-gram (bold letters).

of transduction into one of classification. For the three techniques compared here, the algorithm described in Damper et al. (2005) was used for alignment.

7.2.1 CART: Decision Tree

Since letter-to-phoneme conversion (of aligned strings) is a classification problem, decision trees have long been used for this purpose. Perhaps the earliest such attempt was that of Klatt and Shipman (1982). In this work, we have used the CART algorithm of Black et al. (1998), where CART stands for *Classification and Regression Tree* (see Breiman et al. 1984, p.ix). The CART approach uses feature data to predict class membership. In this case, the feature data are the letters in a fixed-length context window and the class is the phoneme corresponding to the central, 'target' letter. Since the letter and phoneme strings are aligned, a one-to-one correspondence between the target letter and the class (i.e., the corresponding phoneme) can be assumed.

The first step in building a decision tree is to create from each word of the training data a set of learning vectors containing n-grams (context windows consisting of strings of n letters, where n is odd) and the corresponding phoneme for the middle letter of each. As an example, Figure 7.1 shows all 7-gram learning vectors for the name AARDEMA whose pronunciation is /AA R D EH M AH/ (according to the CMUDICT phoneme set used in this work—see Appendix A).

Next, the learning vectors for each word in the training dataset are fed as the input into CART to create the decision tree. In the training process, predictions about the phonemes corresponding to earlier letters in the word can be used to make decisions about the phoneme corresponding to the current, target letter; this is called phonemic feedback. Thus, there are two parameters in CART that we need to fine-tune, namely the width n of the context window and number of previous phonemes, P, to be used in phonemic feedback.

7.2.2 Table Look-Up I: A Simple Look-Up Procedure

This method was proposed by Weijters (1991) who drew the conclusion that his simple look-up procedure is superior to NETtalk (Sejnowski and Rosenberg 1987). The first step is to create a look-up table from a training set containing n-grams (string of n letters, n odd), their corresponding phoneme(s) for the middle letter of n-gram, and their frequencies in the training data. One n-gram is produced for each letter of the input, with each letter serving in turn as the centre of the n-gram. To obtain the pronunciation for an input string, we search for the *closest fit* of n-grams, i.e., those with the highest matched value between the n-grams of the input and those of the pre-compiled look-up table.

```
Weight[1..7] := 1, 4, 16, 64, 16, 4, 1
MatchValue := 0
for i := 1 to 7 do
begin
   if windowL[i] = windowT[i] then
     MatchValue := MatchValue + Weight[i]
   end if
end for
```

The matched value indicates the similarity between a n-grams of an input and a n-gram of a look-up table. The matched values is calculated as in the example pseudocode (i.e., $n = 7$, so-called heptagram) above.

Here, the heptagram in a look-up table is referred to as `windowL` and the heptagram in an input is referred to as `windowT`. The nth letter in `windowL` is referred to as `windowL[n]`.

To clarify the look-up procedure, we explain by giving an example of the heptagram *ENDROTH* from the word ABENDROTH. The heptagram *INDROTH* from the word LINDROTH is found to be the closest fit. The algorithm determines the MatchValue between the heptagrams *ENDROTH* and *INDROTH* to be:

$$0 + 4 + 16 + 64 + 16 + 4 + 1 = 105$$

In the table the middle grapheme of *INDROTH* has been transcribed as the phoneme /R/. Thus, the middle grapheme in *ENDROTH* is assigned to the phoneme /R/.

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

Weijters actually used a wide variety of weight sets and window sizes, but results did not differ too much for the different choices. Table 7.1 is an example of 15 different weight

Window positions and corresponding weights											% phonemes
1	2	3	4	5	6	7	8	9	10	11	correct
0	0	0	0	0	1	0	0	0	0	0	51.25
0	0	0	0	1	4	0	0	0	0	0	69.39
0	0	0	0	0	4	1	0	0	0	0	69.40
0	0	0	0	1	4	1	0	0	0	0	83.77
0	0	0	1	4	16	4	0	0	0	0	86.19
0	0	0	0	4	16	4	1	0	0	0	88.67
0	0	0	1	4	16	4	1	0	0	0	89.68
0	0	0	1	4	16	4	2	0	0	0	89.86
0	0	1	4	16	64	16	4	1	0	0	90.52
0	0	1	4	16	64	16	5	1	0	0	90.60
0	1	4	16	64	256	64	17	4	0	0	90.45
0	0	4	16	64	256	64	16	4	1	0	90.81
0	0	4	16	64	256	64	17	4	1	0	90.82
0	0	16	64	256	1024	256	64	16	4	1	90.86
0	0	16	64	256	1024	256	65	16	4	1	90.88

TABLE 7.1: The percentage of phonemes correct for different values of 15 weight sets on a 11-gram window.

sets on a 11-grapheme window size, reported with the percentage of phonemes correct on the test data.

7.2.3 Table Look-Up II: Table Look-Up with Defaults

Van den Bosch and Daelemans (1993) describe a simple table look-up procedure with some default tables that are invoked in the case of matching failure so as to improve generalisation ability. During table construction, all unambiguous one-to-one letter-to-phoneme mappings are found and stored in the 0-1-0 subtable. Then, the width of the letter window is expanded on the right by one character, and all unambiguous 0-1-1 patterns found and stored in the 0-1-1 subtable, excluding those patterns already in the 0-1-0 subtable. Then, the window width is expanded on the left by one character and the procedure repeated. The process of expanding the window on right or left and storing all the patterns that have not been stored in the earlier table continues until all patterns in the training set are compressed in the look-up table. In this work, this occurs with a 10-1-10 window. Additionally, two default tables are assembled to provide generalisation ability. The first default table, referred to as the *best-guess* table, contains all occurring 1-1-1 patterns and their most frequently occurring phonemic mapping. The second table, referred to as the *final-guess* table, contains all letters and their most frequently occurring phonemic mappings.

The conversion algorithm starts by searching for a matching letter pattern for each letter of an input word in the 0-1-0 subtable. Note that, if found, this is guaranteed to

Left context	Focus letter	Right context	Target phoneme	Subtable
+	U	P	/AH/	best guess
	P	D	/P/	0-1-1
	D		/D/	final guess
D	I	K	/IH/	best guess
	K	E	/K/	0-1-1
DIK	E	+++	/-/	3-1-3

TABLE 7.2: Example of the retrieval for the pronunciation of the word UPDIKE.

be unambiguous. If no pattern is matched, each (single-letter) pattern is extended to a 0-1-1 pattern and the 0-1-1 subtable is then searched. This is repeated until a matching pattern with a minimal extension is found and the corresponding letter-phoneme mapping is returned. If no match can be found, the best-guess table is scanned to return the 'best' mapping. If table look-up fails again, the default phoneme for that letter from the final-guess table is returned. Finally, all phonemes are concatenated to create the pronunciation of the word.

From the example in Table 7.2, an unambiguous pattern with minimal context is searched for each focus letter. Plus symbols represent spaces and blank spaces mean no left or right context. In the case of the first letter (U) and the fourth letter (I), no pattern in subtables is matched. Thus, the algorithm searches for 1-1-1 pattern in the best-guess table. When the best-table retrieval fails in the third letter, the corresponding phoneme of 0-1-0 pattern for letter D in the final-guess table is returned.

7.3 Re-Implementation Details

The re-implementation of three data-driven methods: PbA, Table look-up I, and Table look-up II, was programmed in Python version 2.3.4 running on Windows XP using a PC with 2.66 GHz Intel Pentium 4 and 1 GB RAM. For the CART method only, the source code was provided and run on a different computer (see Section 7.3.2). The re-implementation details of three methods are described below, including how to run a CART program.

7.3.1 PbA

The re-implementation of multi-strategy PbA was programmed in Python 2.4, as details in Chapter 3. Prior to selecting a subset of pronunciations for subjective testing in the next chapter, pronunciations were produced for *all* names in the dictionary. This served as the objective evaluation. It is, of course, essential that these pronunciations are for 'unknown' words (i.e., 'unseen' in the sense of not being available to the analogical

inferencing process). For PbA, this is easily achieved using the very simple leave-one-out strategy.

7.3.2 CART

The source code for CART was developed by Vincent Pagel (Faculte Polytechnique de Mons) and Kevin Lenzo (Carnegie Mellon University, USA) (Pagel 2005). The programs were implemented in C and Perl. The package, which was compressed in a file `ID3_GPL050913.tar.gz`, consisted of two parts: programs for alignment and programs to build and run a decision tree from an aligned corpus. Since the data we used were already aligned using the algorithms of Damper et al. (2004), the second part of the package only was used here. We run these programs on RedHat Linux 6.2 using a PC server with 650 MHz Intel Pentium III and 512 MB RAM. The vector file was automatically built from an aligned dictionary using the file `ali2vec_id3.pl`. Then, this vector file was fetched into another program to build a decision tree. To run a decision tree, the package provided a program to evaluate using 10-fold cross validation, namely `tenfold_crossvalid.pl`. There are three parameters to fine-tune the performance, which are the number of letters on the left and right context, and phonemic feedback (as mentioned in the previous section).

Again, it is obviously important that the pronunciations used in the subjective tests are 'unseen'. To ensure this, we trained and tested the trees using leave-one-out on the entire set of dictionary words (52,911 names, see Section 7.4 below). This served as the objective evaluation of CART. To evaluate using leave-one-out, the original 10-fold cross validation file was modified. Leave-one-out was enormously expensive computationally (approximately one month elapsed time) since the trees had to be rebuilt over 50,000 times. This was done with parameters $n = 9$ and $P = 3$, which were the best performing in a prior 10-fold cross-validation.

7.3.3 Table Look-Up I

The re-implementation of a Table look-up I was programmed, following the paper of Weijters (1991). In this work, a heptagram structure (window size, $n = 7$) was used with the weight vector of (1, 4, 16, 64, 16, 4, 1). The other sets of weight vectors and n-grams structure were not used due to the fact that the results presented in his work did not significantly differ.

To evaluate the performance using leave-one-out, a note was made during table compilation of those heptagrams which were unique to a particular word and that word was stored with the heptagram. Then, when finding a pronunciation for a particular word, heptagrams unique to that word were removed from the table look-up table. Frequencies

of non-unique heptagrams of that particular word were left unaltered because of the computational difficulty of adjusting them to account for the separate removal of over 50,000 entries. This was done for all 52,911 names in the dictionary, and also served as the objective evaluation of Table look-up I.

7.3.4 Table Look-Up II

The re-implementation of a Table look-up II was programmed, following the paper of van den Bosch and Daelemans (1993). In the table-construction step, we found that there were no unambiguous patterns stored in the 0-1-0 subtable. All the patterns in the training are compressed in the look-up table at 10-1-10 window. Figure 7.2 displays the magnitudes of the subtables in this model. The 2-1-2 subtable stored the majority of the unambiguous patterns and the number of stored patterns gradually decreases when window widths were extended.

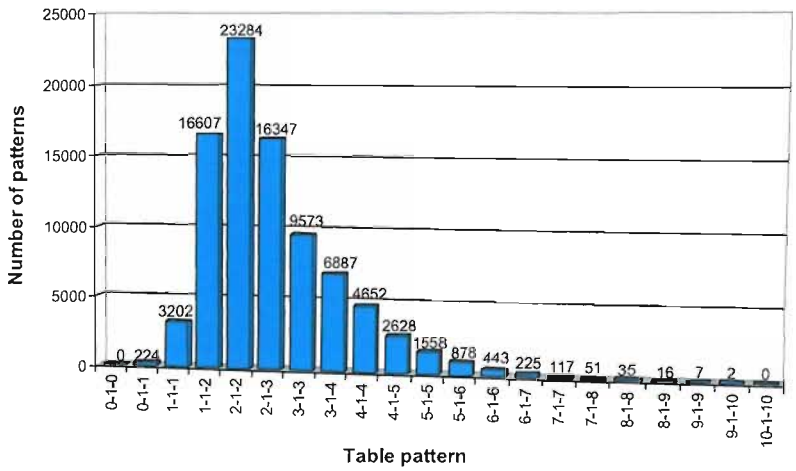

FIGURE 7.2: Table magnitudes of look-up subtables in re-implementation of van den Bosch and Daelemans (1993).

As with the other data-driven methods, it was important that pronunciations corresponded to 'unseen' words. This was achieved as follows. During table creation, the specific words used to derive each pattern (i.e., table entry) was noted. If the pattern was unique, in the sense that only one word exhibited that pattern, this pattern was ignored in deriving a pronunciation for that unique word. This was the only adjustment made. Strictly, the frequency of the mappings ought also to be reassessed, but for simplicity (and because we believe it had little effect on results) this was omitted. Again, this was done for all 52,911 names in the dictionary, and also served as the objective evaluation of Table look-up II. Note that no unambiguous 0-1-0 patterns were actually found in CMUDICT.

	% words	% phonemes
PbA	68.35	94.31
CART	61.45	90.67
Table look-up I	58.55	91.26
Table look-up II	61.89	92.14

TABLE 7.3: Evaluation of pronunciations of 52,911 proper names by four automatic methods in terms of words correct and phonemes correct.

7.4 Material

The test data were a list of 52,911 proper names from CMUDICT, which were previously described in Chapter 6. The listed pronunciations were taken to be the correct 'gold standard' for all four data-driven methods.

7.5 Experimental Results

In this section, we present the results obtained with the four automatic pronunciation approaches as described in the previous section. The results were reported in terms of words correct and phonemes correct. The performances of all four methods were evaluated using leave-one-out. Taking the dictionary pronunciations as correct, results were obtained by scoring the automatically-derived pronunciations against the pronunciation in the name list. As Damper et al. (1999) mentioned, words correct is a more stringent measure of pronunciation accuracy and should be used in strong preference to phonemes correct. Thus, the percentage of word accuracy was used for analysis and discussion.

As can be seen in Table 7.3, PbA achieved the highest percentage of words correct and percentage of phonemes correct. The difference between word accuracy for PbA and for the next best method (Table look-up II) was enormously statistically significant (binomial test, $z \sim 30.74$, $p \sim 0$). As expected, Table look-up II achieved enormously better performance than Table look-up I because of the extension to include default tables. The difference between word accuracy for Table look-up II at 61.89% and for CART at 61.45%, was apparently small, but is in fact statistically significant (binomial test, $z \sim 2.07$, $p \sim 0.04$) when dealing with tens of thousands of words. Note that the phonemes correct measures do not align with words correct. Table look-up I achieved better phonemes correct score compare to CART but was far poorer in terms of words correct (binomial test, $z \sim 13.54$, $p \sim 0$). This indicated that phoneme errors are not independently distributed across words, and is another reason for preferring words correct as our measure of effectiveness. Note finally that, for simplicity, stress assignment was ignored in this objective evaluation.

7.6 Conclusion

We have compared four automatic, data-driven methods for proper name pronunciation within a text-to-speech system for English, namely: pronunciation by analogy (PbA), decision trees (CART), and the table look-up method by Weijters (Table look-up I), and the table look-up method by van den Bosch and Daelemans (Table look-up II). These four data-driven methods are intended as candidates for the back-up strategy used when dictionary look-up fails in a practical TTS system. The comparison was primarily objective, on 52,911 names in CMUDICT. The best result was 68.35% words correct, achieved using PbA. This was significantly far better than the results of the other three methods. From best to worst in terms of words correct, the three methods are ordered Table look-up II, then CART, then Table look-up I.

This experiment has contributed to an understanding of why PbA is better on this task. The results are a good support to the hypothesis of Daelemans et al. (1999), mentioned earlier in Chapter 3, that "keeping full memory of all training instances is at all times good idea in language learning". Furthermore, it is a good empirical evidence of the advantage of explicit analogy. The other three pronunciation methods are implicit analogy, in which phonological knowledge from dictionary is implicitly extracted, and converted into a knowledge-based representation i.e., decision trees and tables in this case. During the learning process, implicit analogy tends to compress the redundancy into small descriptions of the original data. Some exceptions or low-frequency data may be discarded since the method tends to generalise and abstract the training data. The exceptions of proper names seem to be rare, the rare events are characterised as belonging to the LNRE class of distributions. To obtain high accuracy in learning NLP tasks, carefully handling of the LNRE phenomenon is essential (Marchand and Damper 2007). To generate a pronunciation of an unknown word, PbA implicitly exploits the knowledge from the dictionary itself, while the other three methods use their compress representation. Thus, the exceptions are kept at all times in case of PbA. That may be the reason why PbA performance is much better than that of the other methods. Moreover, PbA does not use a fixed-size window on the input text. Instead, substring matching in PbA uses variable-size chunks. We believe that this is another advantage, since different-size chunks will be appropriate in different transcription situations.

It is also interesting to assess the acceptability of the pronunciations to potential users of a TTS system. Therefore, subjective evaluation was carried out. The results is presented in the next chapter. To reduce the amount of test data for subjective testing, Table look-up I was not used in subjective evaluation, because it achieved the lowest words accuracy and shared a similar concept to Table look-up II.

Chapter 8

Subjective Evaluation

8.1 Introduction

Numerous studies have proposed various approaches to the problem of proper name pronunciation and have reported the results in different ways. Most studies evaluate the performance objectively in terms of word accuracy, by comparing the pronunciations generated by their models with those in some data set taken as a 'gold standard'. However, the pronunciation should also be acceptable to users of the TTS system, which obviously includes members of the general public with minimal exposure to synthetic speech. Thus, subjective evaluation is also important, particularly for proper names, which can be pronounced differently depending on the linguistic background of the speaker and other cultural factors. Because subjective evaluation is difficult compared to objective evaluation, and cannot deal with large numbers of names, there have been few attempts to do this (but see Font Llitjós and Black 2002 who have conducted informal subjective evaluation of proper name pronunciations over the internet).

In this chapter, the acceptability of the differently-produced pronunciations is assessed by potential users of a TTS system. The comparison here is a subjective counterpart to the objective evaluation introduced in the previous chapter. Four different methods of proper name pronunciation for English TTS synthesis is compared. The first uses a dictionary of manually-supplied pronunciations, containing a list of proper names from CMUDICT. The dictionary is referred to as a 'method' in this chapter because it is generally the primary strategy in any practical TTS system. The remaining three methods are different data-driven approaches that use the dictionary of (known) proper names to infer pronunciations for unknown names; these are candidates as secondary or back-up strategies for those cases where dictionary matching fails. The three candidate techniques studied are: PbA, CART, and the table look-up method II. The description of the dictionary and these techniques have already been presented in the previous chapter.

Table look-up I, from the previous chapter, was not chosen for several reasons. First, it achieved the lowest word accuracy among the four data-driven methods presented in Chapter 7 and, second, we need to reduce the number of test data used for test stimuli. Both table look-up methods studied in the objective tests share the same concept. So, it was not felt necessary to include both.

The remainder of this chapter is structured as follows. Section 8.2 details the listening tests that were performed to evaluate the acceptability of the pronunciations produced by the dictionary and by the three data-driven techniques, including the means of selecting a reasonable number of names to test from among the large number ($> 50,000$) available in the dictionary. Section 8.3 presents the results of the subjective evaluation. Section 8.4 discusses and concludes.

8.2 Experimental Design

The listening tests were designed to assess the acceptability of the proper name pronunciations produced by the four methods (manually-specified via CMUDICT, hereafter 'CMU', and the three different data-driven methods) to potential users of a text-to-speech system for English. Although separately evaluated as such, we do not intend the dictionary pronunciations to be considered as exactly equivalent to the data-driven pronunciations for two reasons. First, dictionary look-up would be used in a practical system in conjunction with one or other data-driven method (in the way of a primary and a back-up strategy). Second, the data-driven methods use the dictionary as a knowledge base from which to make inferences about an acceptable pronunciation of 'unknown' names, so the two (dictionary and automatic inference) are clearly neither equivalent nor independent.

8.2.1 Selection of Test Stimuli

It is obviously not practical to expect listeners to rate all 52,911 pronunciations in our dictionary, so some principled way of selecting among the large number of names to produce a much smaller subset suitable as test stimuli is required. This is not especially easy to do for two main reasons:

1. In spite of the CMU pronunciations being taken as the 'gold standard' for the automatic inference of pronunciations using the three data-driven methods, we did not wish to assume that the dictionary pronunciations are actually correct. (There are, in fact, some very obvious errors in the dictionary.) The CMU method is to be assessed via the listening tests on the same basis as PbA, CART and TLU. This means that we cannot identify 'errors' in pronunciation as such. Rather,

Pattern	Number of cases
All pronunciations same, zero disagreement	$C(4,0) = 1$
Three pronunciations same, remaining 1 different	$C(4,1) = 4$
Two pronunciations same, remaining 2 different	$C(4,2) = 6$
Two pronunciations same, remaining 2 same	$\frac{C(4,2)}{2} = 3$
Zero pronunciations same, zero agreement	$C(4,4) = 1$
TOTAL	15

TABLE 8.1: Possible patterns of disagreement/agreement in pronunciations produced across the four methods. The number of "two pronunciations same, remaining 2 same" cases is reduced by a factor of 2 because of symmetry.

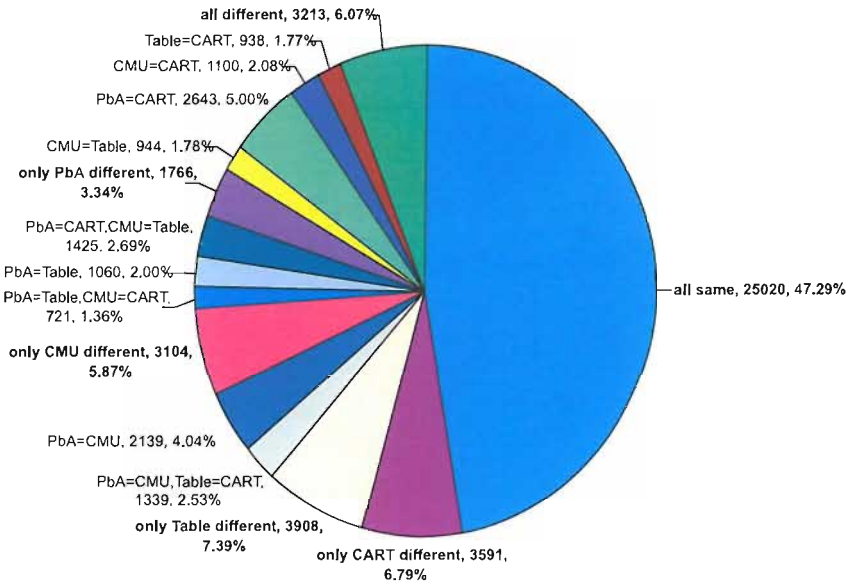

FIGURE 8.1: Pattern of (dis)agreement across the four methods.

pronunciations are to be treated as more or less acceptable to potential users. So we cannot use any notion of error in selecting stimuli for presentation to listeners.

2. Since we have four methods of arriving at pronunciations, the number of possible patterns of disagreement/agreement across the four is quite large. In fact, consideration of the combinatorics shows that there are 15 different possible patterns (Table 8.1). Figure 8.1 shows the proportion of pronunciations exhibiting each of these different possible patterns of (dis)agreement. If we were to select names as stimuli at random from within the total of 52,911, there would be no control of the pattern of disparity across methods, rendering meaningful analysis of the results virtually impossible. It seems preferable to focus on just one of the patterns of (dis)agreement, i.e., just one of the rows of Table 8.1.

For these reasons, especially 2), we decided to use as stimuli a randomly-sampled subset of those names on which only one of the four methods disagreed. This corresponds

to the second row of Table 8.1 for which there are 4 cases listed, one for each of the 4 methods. This is the only pattern of (dis)agreement giving clear separation of methods. Those pronunciations for names in the sampled subset which were produced by just one of the four methods (the others agreeing) we will henceforth call *one-of-a-kind*. Those pronunciations on which the remaining 3 of the 4 methods (i.e., the rest) agree we will henceforth call *rest*. The reasoning behind this choice is that, given that we have no strong basis for deciding what an error is, name pronunciations on which 3 out of 4 methods agree are likely to be acceptable to listeners, whereas the corresponding *one-of-a-kind* pronunciations are likely to be representative of the worst pronunciations made by the particular method delivering the deviant output.

Thus, we randomly selected 150 names with *one-of-a-kind* pronunciations for each method, a total of 600. For example, we selected 150 names from the 3,591 names for which CART produced a pronunciation at variance with the other three methods, which agreed (see 'only CART different' in Figure 8.1). As well as these 600 names with *one-of-a-kind* pronunciations, we also had the same set of 600 names with a corresponding *rest* pronunciation, a total of 1200 stimuli.

8.2.2 Synthesis

Speech output was by diphone synthesis using Festival, a public domain system intended for speech synthesis research available from `http://www.cstr.ed.ac.uk`. To achieve good pronunciations, it was necessary to syllabify the words and add stress. Syllabification was done using the Festival function `lex.syllabify.phstress(PHONELIST)`. To obtain stress patterns, we passed the spelling patterns through Festival's letter-to-sound rules (Black et al. 1998) and then manually transferred the stress pattern of the output to the 1200 stimuli. The voice used was male KAL.

8.2.3 Test Conditions

Subjects listened to the 1200 synthesised names over headphones in the soundproof room in our laboratory at Southampton. We decided firmly against online internet-based evaluation, as used by Font Llitjós and Black (2002), because of the inherent lack of experimental control with that approach. Subjects saw the names printed on a sheet as well as hearing the synthesised version. They were instructed to rate their opinions on the quality of the pronunciations on a five-point scale, according to the mean opinion score (MOS) (ITU-T Rec.830 1996) as follows:

1 Unacceptable, no one would ever pronounce this name like this

2 Poor, very few people would recognise the name from this pronunciation

3 Acceptable, some people would recognise this name

4 Almost correct, most people would recognise this name

5 Correct, almost everybody would be able to recognise this name

Listeners wrote their ratings next to the printed names. They were given printed instructions at the outset. Subjects were informed that apostrophes in names (e.g., "O'Connell") had been removed. They were further told that they would hear one name every two seconds, and be given a short break at the end of each page of (30) printed names, after which they should press a button to signal that they were ready to continue. They were asked to assess the pronunciations as if they had been produced by a telephone directory assistance service.

The complete experiment took some 70–80 minutes per subject. It was quite demanding of listeners both in time and concentration. We therefore feel that the number of names selected for the experiment (600 *one-of-a-kind* and 600 *rest*) is effectively a practical maximum.

8.2.4 Subject Profiles

Since MOS values generally become more stable as the number of listeners increases, a reasonably large pool of subjects should be used, balanced by the time and cost of testing; Clark (2005) suggests using at least 16 subjects. Accordingly, 24 listeners (15 male, 9 female) took part in this study. All were students at the University of Southampton, and native speakers of British English, aged between 18 and 31 years. Of the 23 subjects who responded to a prior question about their familiarity with synthetic speech (see Appendix C), 2 were "very familiar", 14 were "somewhat familiar", and 7 were "not at all familiar". (Note that one of the 24 subjects omitted to answer this question.) Listeners were paid a small sum (£20 each) for their participation.

Subjects were warned that they would hear 'Americanised' names. We were well aware of the disparity between our subjects' version of English (British) and that of the materials to which they were exposed (American English) but this was unavoidable. There are no corresponding materials easily available for British English, and we did not have access to a pool of American English subjects. We do not believe this had a serious effect; young educated British people generally have considerable exposure to popular American culture.

8.3 Experimental Results

In this section, results in terms of MOS values for *one-of-a-kind* pronunciations and *rest* pronunciations (on which 3 of the 4 methods agree) are presented. Given the justification in Section 8.2.1 for the selection of names to test, it is reasonable to view high MOS values for *one-of-a-kind* pronunciations as indicative of the overall quality of a method, since they are likely to be representative of the worst pronunciations produced by that method. Things are not so simple, however, for the *rest* pronunciations. Here, the pronunciations are representative of the general quality of the 'opposition', i.e., the three competitor methods. So high MOS values for the *rest* pronunciations can be taken to indicate that the corresponding method is problematic, or at least inferior to the opposition. In other words, if a particular method produces pronunciations which are rated highly by subjects but which are different to pronunciations produced by the remaining methods, these latter being rated poorly by listeners, then there is a sound basis for considering this method to be superior to its competitors. It is necessary to emphasise that we are referring here to relatively high/low values *across* methods. We expect the *one-of-a-kind* MOS values to be relatively lower than the *rest* values *within* methods, just because the former are likely to be representative of the worst pronunciations produced, as argued previously.

Before describing the results quantitatively, some example pronunciations are highlighted for purposes of illustration. Table 8.2 shows the worst and best pronunciations, both *one-of-a-kind* and *rest*, in terms of obtained MOS values for each of the four competitor methods. (The reader should note that here the 'mean' is for a particular pronunciation across all 24 listeners.) The CMU phoneme symbols are used here in preference to those of the International Phonetic Alphabet (International Phonetic Association 1999). Not only do these examples give an idea of the range of quality of pronunciations produced, there are also some interesting trends evident. The expectation that the *one-of-a-kind* MOS values should be generally lower than the *rest* values within methods is confirmed, at least for these best/worst pronunciations. The relatively high MOS values for *one-of-a-kind* coupled with relatively low values for *rest* pronunciations for CMU indicate that the dictionary pronunciations are generally superior. This is to be expected, since the dictionary forms the 'gold standard' knowledge base from which the data-driven methods infer pronunciations. There is no obvious basis on which the inferential process could *improve* on the dictionary pronunciations.

(a) Worst *one-of-a-kind*

Method	name	*one-of-a-kind* pron.	MOS	*rest* pron.	MOS
CMU	Freda	F R EH D	1.63	F R EH D AH	2.88
PbA	Czech	CH AH HH	1.21	CH EH K	4.92
CART	Joey	JH AA IY	1.29	JH OW IY	4.75
TLU	Borchard	B ER AA R D	1.58	B ER SH AA R D	4.00

(b) Best *one-of-a-kind*

Method	name	*one-of-a-kind* pron.	MOS	*rest* pron.	MOS
CMU	Even	IY V AH N	4.88	EH V AH N	3.17
PbA	Hamlet	HH AE M L EH T	4.96	HH AE M L AH T	4.92
CART	Stasney	S T AE Z N IY	4.83	S T AE S N IY	4.50
TLU	Imperato	IH M P ER R AA T OW	4.83	IH M P ER AA T OW	4.38

(c) Worst *rest*

Method	name	*one-of-a-kind* pron.	MOS	*rest* pron.	MOS
CMU	Tutor	T UW T ER	4.67	T AH T ER	1.50
PbA	Dupre	D AH P R	1.96	D AH P ER	2.33
CART	Goucher	G AW CH ER	3.96	G AW K ER	2.42
TLU	Birr	B R	2.25	B ER	2.25

(d) Best *rest*

Method	name	*one-of-a-kind* pron.	MOS	*rest* pron.	MOS
CMU	Fossett	F AA S AH T	4.83	F AA S EH T	4.83
PbA	Pandora	P AA N D AO R AH	4.46	P AE N D AO R AH	4.92
CART	Melanie	M EH L AA N IY	4.79	M EH L AH N IY	4.96
TLU	Terri	T ER R IY	3.67	T EH R IY	4.88

TABLE 8.2: Example pronunciations, showing the worst and best pronunciations, both *one-of-a-kind* and *rest* in terms of obtained MOS values for each of the four competitor methods.

8.3.1 Results for *One-of-a-Kind* Pronunciations

General statistics of MOS values for these pronunciations are listed in Table 8.3. (Note that here the 'mean' is for a particular method across all *one-of-a-kind* pronunciations and all 24 listeners.) MOS values exceed 3.7 (i.e., tending towards 'Almost correct') indicating that all methods gave reasonable pronunciations. As expected, dictionary pronunciations have a higher MOS than any of the automatically-inferred methods. From best to worst in terms of MOS of *one-of-a-kind* pronunciations, the data-driven methods are ordered PbA, then CART, then TLU.

Since our data are not interval data but ordinal and, as a consequence, can not be normally-distributed, and because many statisticians believe that means are an inappropriate measure of central tendency for ordinal data (although the MOS measure already violates this belief), we believe non-parametric statistical analysis (Siegel and Castellan 1988, p.75-83) is appropriate. Specifically, we have used the Wilcoxon signed-rank test.

	Mean	Std. Deviation
CMU	4.0372	0.5443
PbA	3.8319	0.6760
CART	3.7790	0.7285
TLU	3.7305	0.7157

TABLE 8.3: Mean opinion scores for subjective evaluation by 24 listeners of 600 one-of-a-kind pronunciations.

Subject	CMU	PbA	CART	TLU
1	1	2	3	4
2	1	2	4	3
3	1	2	3	4
4	1	2	4	3
5	1=	1=	3	4
6	1	2	4	3
7	1	3	2	4
8	1	2	3	4
9	1	3	2	4
10	1	2	3	4
11	1	3	4	2
12	1	2	3	4
13	1	2=	2=	4
14	1	3	2	4
15	1	2	3	4
16	1	2	3	4
17	1	4	2	3
18	1	2	4	3
19	1	2	3	4
20	1	2	3	4
21	1	4	3	2
22	1	2	3	4
23	1	4	2=	2=
24	1	2=	4	2=

TABLE 8.4: Rankings of the four methods for the one-of-a-kind pronunciation by the 24 subjects according to MOS values. The equals sign ('=') indicates tied rankings.

Table 8.4 shows the order in which each of the 24 listeners ranked the *one-of-a-kind* pronunciations produced by each of the 4 methods according to the corresponding MOS values. It is very striking that all 24 consistently rated the CMU pronunciations first. The data-driven methods have been ranked fairly consistently in the order PbA, CART, TLU; 10 out of 24 listeners ranked the methods in this order. Table 8.5 shows the results for the Wilcoxon signed-rank test applied to these data. The only significant differences in the opinions of the listeners are between CMU and each of the data-driven methods ($p < 0.05$, two-tailed, df = 149), with CMU superior to the others.

	Z	Sig. (two-tailed)
CMU–PbA	−2.535	0.011
CMU–CART	−2.737	0.006
CMU–TLU	−3.268	0.001
PbA–CART	−0.366	0.715
PbA–TLU	−0.974	0.330
CART–TLU	−0.341	0.733

TABLE 8.5: Results of Wilcoxon signed-rank test for one-of-a-kind pronunciations.

	Mean	Std. Deviation
CMU	4.0036	0.6396
PbA	4.1975	0.4685
CART	4.2800	0.4628
TLU	4.2778	0.3937

TABLE 8.6: Mean opinion scores for subjective evaluation by 24 listeners of the 600 rest pronunciations.

8.3.2 Results for *Rest* Pronunciations

General statistics of MOS values for these words are listed in Table 8.6. (The reader should note that here the 'mean' is for a particular method across all *rest* pronunciations and all 24 listeners.) As explained and justified earlier, we expect the MOS values to be generally higher for *rest* pronunciations than for *one-of-a-kind* pronunciations (because the latter are representative of the worst pronunciations produced by a given method), whereas relatively high MOS values for *one-of-a-kind* pronunciations and relatively low values for *rest* pronunciations for a given method are taken to be indicative of the quality of that method. As the table shows, MOS values generally exceed 4 (i.e.,'Almost correct') and there is indeed a strong tendency towards higher MOS values than for *one-of-a-kind* pronunciations.

Comparing Table 8.6 with Table 8.3 earlier, we see (as expected) an almost inverse or 'complementary' pattern of results. The CMU *rest* pronunciations have the lowest MOS value, with the remaining three data-driven methods ordered from lowest to highest as PbA, TLU, CART with the latter two almost indistinguishable. Figure 8.2 shows a composite plot of the variation of MOS across the four methods for both *one-of-a-kind* and *rest* pronunciations, making the 'complementary' nature of the results very clear. In interpreting this figure, readers should recall that *one-of-a-kind* scores are considered to be representative of the worst pronunciations produced by a method whereas *rest* scores are considered to be representative of the quality of pronunciations produced by the three competitor methods. One interesting feature is that listeners rated the CMU *one-of-a-kind* and *rest* pronunciations as effectively indistinguishable, meaning that the worst dictionary pronunciations are comparable in acceptability to some form of average of the three data-driven methods. This is perhaps understandable in hindsight as the latter take the dictionary as their 'gold standard' knowledge source for automatic

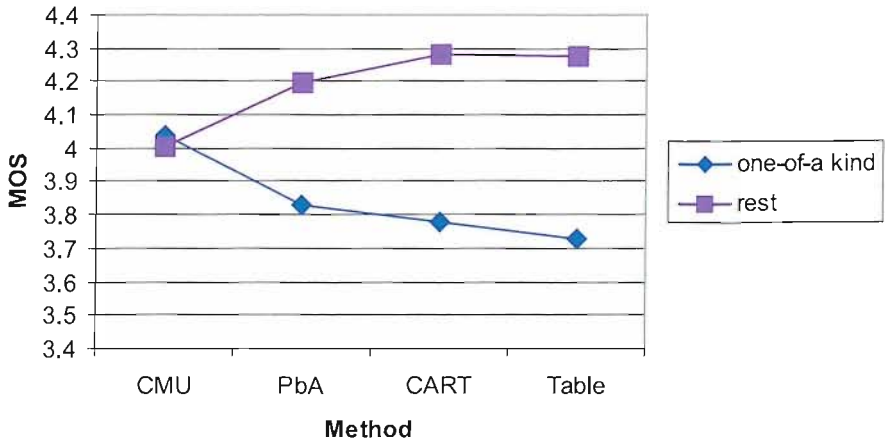

FIGURE 8.2: Composite pattern of MOS values across the four methods for both one-of-a-kind and rest pronunciations. Scores for *one-of-a-kind* words for a method are taken to be indicative of the poorest pronunciations produced by that method. Scores for *rest* words for a method are taken to be indicative of the general quality of pronunciations produced by the three competitor approaches to that method. See text for further explanation.

inferencing. We also take this as indirect evidence of the quality and suitability of the dictionary as a 'gold standard'.

Table 8.7 shows the order in which each of the 24 listeners ranked the *rest* pronunciations produced by each of the 4 methods according to the corresponding MOS values. As expected, these rankings are almost a complete reversal of those for the *one-of-a-kind* pronunciations in Table 8.4.

Table 8.8 shows the results for the Wilcoxon signed-rank test applied to these data. There are highly significant differences between CMU and each of the data-driven methods, with CMU superior to the other techniques ($p < 0.05$, two-tailed test, df $= 149$), and marginally significant differences between PbA and the remaining two data-driven methods ($p \sim 0.1$). This is taken as suggestive evidence for the superiority of PbA as a back-up strategy.

8.4 Conclusion

Four methods for proper name pronunciation within a TTS system for English were compared. These were (1) a manually-compiled dictionary (method CMU) and (2) three automatic, data-driven methods for proper name pronunciation, namely: pronunciation by analogy (PbA), a decision tree method (CART), and a table look-up method (TLU). These four methods are not intended to be viewed as entirely free choices for deployment

Subject	CMU	PbA	CART	TLU
1	4	3	2	1
2	4	3	2	1
3	4	3	1	2
4	4	3	2	1
5	4	3	1	1
6	4	3	2	1
7	4	1	2	3
8	4	3	2	1
9	4	1	3	2
10	4	3	1	2
11	4	3	2	1
12	4	3	2	1
13	4	3	1	2
14	4	3	2	1
15	4	3	2	1
16	4	3	1	2
17	4	3	2	1
18	4	2	3	1
19	4	3	1	2
20	4	3	1	2
21	3=	3=	2	1
22	3	4	1	2
23	4	1	2	3
24	4	1	2	2

TABLE 8.7: Rankings for the four methods for the rest pronunciations by the 24 subjects according to MOS values. The equals sign ('=') indicates tied rankings.

	Z	Sig. (two-tailed)
CMU–PbA	−2.370	0.018
CMU–CART	−4.566	0.000
CMU–TLU	−4.108	0.000
PbA–CART	−1.620	0.105
PbA–TLU	−1.605	0.109
CART–TLU	−0.099	0.921

TABLE 8.8: Results of Wilcoxon signed-rank test for rest pronunciations.

in a practical TTS system; rather we envisage that the dictionary method will form the primary strategy whereas one of the data-driven methods will constitute the back-up strategy used when dictionary look-up fails. The comparison was primarily subjective, on the basis of 24 listeners' opinions of the acceptability of so-called *one-of-a-kind* and *rest* pronunciations. The former, in which one particular method of the four produced a pronunciation which differed from that produced by the other three, was expected to be representative of the worst pronunciations that that particular method would produce. For each *one-of-a-kind* pronunciation, there is a corresponding so-called *rest* pronunciation, being that produced by the remaining three competitor methods. These

were expected to be generally better (in terms of acceptability) than the *one-of-a-kind* pronunciations, and also representative of the quality of pronunciations that competitor methods could produce. So a pattern of relatively high *one-of-a-kind* scores and relatively low *rest* scores is taken as an indication of high performance for a given method.

Listeners rated *one-of-a-kind* pronunciations somewhere around 'Almost correct', with an MOS close to 4, or just below for all four methods, whereas *rest* pronunciations were rated somewhere around 'Almost correct' or just above (see Figure 8.2). Wilcoxon signed-rank tests showed highly significant differences between CMU and the data-driven methods, for both *one-of-a-kind* and *rest* words. The direction of these differences was such as to show that the CMU pronunciations were markedly superior. This was to be expected as the dictionary forms the knowledge base for automatic inference of pronunciations by the data-driven methods. Hence, there is no reason to expect the latter to improve systematically on the dictionary pronunciations. For the *rest* pronunciations, there was a marginal superiority of PbA over the CART and TLU methods. This mirrored the objective evaluation. No such superiority was seen for the *one-of-a-kind* pronunciations.

The issue for the assessment methods probably opens the debate whether subjective testing is sensitive enough to differentiate between systems. As mentioned by Sproat (1997, p. 230), a listener usually comments on speech output in the form of rating scales such as MOS in subjective assessment. Thus, we asked listeners to judge the acceptability of the pronunciations on a five-point MOS scale. The phonetic transcriptions produced by each method were piped through Festival, producing waveforms for the listening test. Pronunciation methods were hidden from the subjects. We have 24 subjects in this experiment, which is a reasonable number to make MOS values more stable. The results showed that our subjective testing failed to find a strong distinction between the three data-driven methods. It is an open question whether this is because the methods are subjectively indistinguishable or because our testing methodology was simply insufficient to reveal a difference. The fact that the dictionary pronunciation emerged as superior suggests that the testing methodology has at least reasonable discriminative power.

Chapter 9

Conclusions

Automatic pronunciation of unknown words is a hard problem of great importance in speech technology. Over the past decades many researchers have tried to predict word pronunciations from the spelling using a wide variety of methods. The initial attempts were to create LTS rules by expert linguists. As the storage capabilities of a computer increase, dictionary look-up methods have been proposed as a primary strategy. A secondary strategy is required when finding a word in the input which is not in the dictionary. A data-driven approach, as studied in this thesis, is an alternative approach that is increasingly being used as the method of choice in TTS applications.

This thesis focuses on a data-driven approach namely pronunciation by analogy (PbA) for generating the pronunciation of unknown words. To date, many variants of PbA have been proposed and evaluated, mostly for common words in English. PbA is selected for this study because of its well-documented superior performance in the Damper et al. (1999) evaluations. The goal of this thesis is to study PbA performance on many important aspects for use in speech synthesis applications. The issue of proper name pronunciation is focused on for the most part. The multilingual aspect of PbA is also studied. This chapter summarises the work presented in this thesis and proposes some future directions.

9.1 Summary of Work

As the classical version of PbA (Dedina and Nusbaum 1991) mentions, there are many advantages of using PbA in text-to-phoneme conversion. First, it eliminates the need for expert linguists to generate LTS rules that may contain errors. Second, it is a language-independent approach; by changing the dictionary, PbA can be used with different languages. The performance of PbA depends mostly on the degree of consistency of relations between orthographic and phonemic patterns. This method may solve

the problem of proper name pronunciation. In this thesis, the aim is to explore the performance of PbA in respect of the abilities referred above. Other aspects of PbA that may help to improve performance were also studied.

Chapter 1 introduced the principles of speech synthesis systems and text-to-phoneme conversion. The problem of automatic pronunciation of unknown words, particularly proper names, was emphasised.

The history of text-to-phoneme conversion techniques was reviewed in Chapter 2. These techniques were categorised and presented into three main approaches. A data-driven approach, which is the inspiration of the PbA method, was mainly focused upon. Pronunciation models for proper names were also reviewed.

In Chapter 3, PbA was reviewed in full detail with examples of each process involved. PbA exploits the phonological knowledge implicit in the system dictionary of known words to infer a pronunciation for an unknown word by computing different ways of assembling the input word from fragments of partially-matching letter substrings and their corresponding partial pronunciations, and choosing between these candidate pronunciations according to some objective criterion. Many variants of PbA were presented and discussed with respect of their performance. The main differences of each variant were also reviewed.

The problem of syllabification was introduced in Chapter 4. A review of previous work on automatic syllabification was summarised. The concept of syllabification by analogy (SbA) was introduced. The results when applying a series model of syllabification and pronunciation, (S+P)bA, with proper names was presented and discussed.

Chapter 5 fulfilled the multilingual potential of PbA. The difficulty of text-to-phoneme conversion on different languages was addressed. PbA was evaluated with 7 European languages using 12 different lexicons. The relationship between transcription accuracy and orthography in these seven languages was studied. As the size of dictionary is an important issue when employing PbA in a TTS system, the relationship between the size of lexicon and the word accuracy obtained were also presented and discussed.

Unknown words typically include proper names or neologisms that have not been yet listed in the lexicon. The issue of classification of unknown words was addressed in Chapter 6. In previous work, PbA has been evaluated with single 'type' of dictionary; that is as researchers assumed that unknown words tend to be neologisms, so the lexicon of common words has been used only. If they focused on proper name pronunciation, a lexicon of proper names would be used only. Therefore, the effect of lexicon composition was studied to investigate the performance of PbA when inferring pronunciations with the lexicon that contains the same or different or mixture types of words.

This thesis compares a number of pronunciation modelling methods on a proper name dataset. In Chapter 7, four different approaches were evaluated objectively, in which

inferred pronunciations were compared with 'gold standard' dictionary pronunciations. The dictionary itself and 3 of those 4 approaches were evaluated subjectively, in which listeners rated synthesised pronunciation using a 5-point scale, in Chapter 8. The results of objective and subjective evaluation were compared and discussed using statistical tests.

In conclusion, this thesis has extended the problem of automatic pronunciation of unknown words in TTS applications, particularly the word class of proper names, using the PbA approach. The performance of PbA was investigated in various aspects. The experimental results show that PbA can achieve a promising level of word accuracy and is superior to other methods tested on the problem of proper name pronunciation. Finally, this thesis has shown that PbA should become the method of choice in a future TTS applications.

9.2 Future Work

Many important aspects of PbA for TTS applications have been examined. A large part of this thesis has been concerned with the problem of proper names. However, there are some aspects of PbA that should be evaluated further. Of these follow-up studies, the most interesting suggestions are as follows:

- **Stress assignment.** The importance of lexical stress varies across languages. In English, it has been found that stress assignment has a serious effect on intelligibility (Slowiaczek 1990). Many researchers have discovered that including stress in letter-to-phoneme conversion helps to improve the accuracy and naturalness of pronunciation. Thus, lexical stress assignment is required in a TTS system to produce a reasonable pronunciation. In our listening tests, Festival's LTS rules were used for stress assignment in all cases, but it would perhaps be better to use a stress-assignment method particular to each pronunciation technique, e.g., manually-assigned for the dictionary, inferred by analogy for PbA, etc., not least because using a common stress assignment for the different methods will contribute to similarity between the three data-driven methods.

- **Automatic syllabification.** syllabification were included in PbA and evaluated the performance with proper names in Chapter 4. However, the results of the series model, which uses both PbA and SbA, show that this syllabification and pronunciation model is inferior to the simple PbA. A possible way to improve the performance of PbA is to include syllabification manually. The comparison of automatic syllabification algorithms would also be worthwhile to investigate, to see which algorithms can improve PbA relative to our standard model. The listening tests should be performed again to evaluate the performance after incorporating syllabification and stress.

- **Alternative ways of selecting stimuli.** The procedural difficulties in selecting word pronunciations to use as stimuli for listening tests have been described in Chapter 8. The experimental design adopted here, which used *one-of-a-kind* and *rest* pronunciations, is by no means the only one possible, nor is it obviously the best. Hence, future work should explore other ways of selecting stimuli for the listening tests. It would also be worthwhile to test a wider range of methods (perhaps including proprietary rules if we can gain access to a set), although the high costs of subjective testing and the practicalities of limiting test times to reasonable durations means that we can never hope to compare large numbers of different approaches.

- **Tests with other languages.** More diverse languages and multiple dictionaries for each language should be studied, if the databases are available. In this thesis, only 7 European languages have been studied. It would be interesting to find out how PbA works in languages other than these, particularly those that do not use roman orthography for their writing system. This might confirm the effect of lexicon size in PbA across all language and maybe clarify the relation between the depth of orthography of a language and the transcription accuracy obtained.

- **Computational requirement.** Since this research has not been concerned with the issue of processing time, this implementation of PbA, which was developed in Python, may not be appropriate for a real world application that needs to respond quickly. Possible ways to reduce the computational requirement for a real-time TTS applications include implementing in C/C++ programming language. The comparison to the other data-driven methods in terms of the processing time would be valuable to justify the performance of PbA.

Appendix A

CMU Phoneme Set

CMU symbol	As in ...	IPA
AA	father	ɑ
AE	bat	a
AH	but	ʌ
AO	bought	ɔ
AW	bout	aʊ
AY	bite	aɪ
B	bet	b
CH	chin	tʃ
D	dime	d
DH	this	ð
EH	bet	ɛ
ER	bird	ɜ
EY	bake	eɪ
F	fin	f
G	guess	g
HH	head	h
IH	bit	ɪ
IY	peat	i
JH	gin	ʤ
K	kitten	k

CMU symbol	As in ...	IPA
L	let	l
M	met	m
N	net	n
NG	sing	ŋ
OW	boat	oʊ
OY	boy	ɒɪ
P	pet	p
R	red	r
S	set	s
SH	shin	ʃ
T	test	t
TH	thin	θ
UH	book	ʊ
UW	lute	u
V	vest	v
W	wet	w
Y	yet	j
Z	zoo	z
ZH	leisure	ʒ

Appendix B

NETtalk Phoneme Set

NETtalk symbol	As in ...
a	wad,dot,odd
b	bad
c	or, caught, awe
d	add
e	blade,way
f	farm
g	gap
h	hot,who
i	bee
k	cab,keep
l	lad
m	man,imp
n	and, gnat
o	only,own
p	pad
r	rap
s	cent,ask
t	tab
u	boot,ooze,you
v	vat
w	we
x	pirate, welcome
y	yes
z	zoo,goes
A	ice, height
C	chart,cello

NETtalk symbol	As in ...
D	the,mother
E	many,end,head
G	length,long,bank
I	give, busy, captain
J	jam,gem
K	anxious,sexual
L	evil,able
M	chasm
N	shorten,basin
O	oil,boy
Q	quilt
R	honer,after,satyr
S	ocean,wish
T	thaw,bath
U	wood,could,put
W	out,towel,house
X	mixture,annex
Y	use,feud,new
Z	usual, vision
@	cab,plaid
!	nazi, pizza
#	auxiliary, exist
*	what
^	up, son, blood
+	abattoir, mademoiselle

Appendix C

Subjective Evaluation Form

Image, Speech and Intelligent Systems

PROPER NAME PERFORMANCE EVALUATION

SUBJECT INFORMATION

Name : _____ **Age** : _____

Languages : _____

How familiar are you with synthetic speech? (delete as appropriate).

VERY FAMILIAR / SOMEWHAT FAMILIAR / NOT AT ALL

EVALUATION GUIDELINES

This is an evaluation of a speech synthesis system for proper names.
Please follow the instructions;

1) Listen to pronunciations of the names on the provided list.
2) Rate the pronunciations using the following scales:

1 = Unacceptable, no one would ever pronounce this name like this
2 = Poor, very few people would recognise the name from the pronunciation
3 = Acceptable, some people would recognise this name
4 = Almost correct, most people would recognise this name
5 = Correct, almost everybody would be able to recognise this name

NOTES

- Apostrophes in names (e.g. in "O'Connell") have been removed.
- You will hear one name every 2 seconds.
- There will be a short break (20 seconds) at the bottom of each page.
- You should assess the pronunciations as if they had been produced by a telephone directory assistance service.
- You will receive £10 payment and will be asked to sign a receipt for this amount.

References

Abercrombie, D. (1981). Extending the Roman alphabet: Some orthographic experiments of the past four centuries. In R. E. Asher and E. Henderson (Eds.), *Towards a History of Phonetics*, pp. 207–224. Edinburgh, UK: Edinburgh University Press.

Adamson, M. J. and R. I. Damper (1996). A recurrent network that learns to pronounce English text. In *Proceeding of 4th International Conference on Spoken Language Processing (ICSLP '96)*, Philadelphia, PA, pp. 1704–1707.

Ainsworth, W. A. and B. Pell (1989). Connnectionist architectures for a text-to-speech system. In *Proceedings of 1st European Conference on Speech Communication and Technology (Eurospeech '89)*, Paris, France, pp. 125–128.

Allen, J., S. Hunnicutt, R. Carlson, and B. Granström (1979). MITalk-79: The 1979 MIT text-to-speech system. *Journal of the Acoustical Society of America 65*(S1), S130.

Arciniegas, F. and M. Embrechts (2000). Text-to-speech conversion with staged neural networks. In C. H. Dagli, L. I. Burke, J. Ghosh, and B. Fernandez (Eds.), *Intelligent Engineering Systems through Artificial Neural Networks: Smart Engineering System Design*, Volume 10, pp. 733–738. New York, NY: ASME Press.

Bagshaw, P. C. (1998). Phonemic transcription by analogy in text-to-speech synthesis: Novel word pronunciation and lexicon compression. *Computer Speech and Language 12*(2), 119–142.

Black, A. W., K. Lenzo, and V. Pagel (1998). Issues in building general letter-to-sound rules. In *Proceedings of 3rd European Speech Communication Association (ESCA)/COCOSDA International Workshop on Speech Synthesis*, Jenolan Caves, Australia, pp. 77–80.

Bullinaria, J. A. (1994). Connectionist modelling of spelling. In *Proceedings of the 16th Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, pp. 78–83.

Caseiro, D., I. Transcoso, L. Oliveira, and C. Viana (2002). Grapheme-to-phone using finite-state transducers. In *Proceedings of 2002 IEEE Workshop on Speech Synthesis*, Santa Monica, CA, pp. 215–218.

Cherkassky, V. and F. Mulier (1998). *Learning from Data*. New York, NY: John Wiley.

Church, K. W. (1985). Stress assignment in letter-to-sounds rules for speech synthesis. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, Chicago, IL, pp. 246–253.

Clark, A. (2005). Tech note: Voice quality measurement. Last accessed on 20 September 2007 from URL `http://www.tmcnet.com/tmcnet/articles/2005/voice-quality-measurement-voip-alan-clark-telchemy.htm`.

Clements, G. N. (1988). The role of the sonority cycle in core syllabification. Working Papers of the Cornell Phonetics Laboratory, WPCPL No. 2, Research in Laboratory Phonology, Cornell University, Ithaca, NY.

Coltheart, M. (1978). Lexical access in simple reading tasks. In G. Underwood (Ed.), *Strategies of Information Processing*, pp. 151–216. New York, NY: Academic Press.

Daelemans, W. and A. van den Bosch (1992). Generalization performance of backpropagation learning on a syllabification task. In *Proceedings of 3rd Twente Workshop on Language Technology*, Enschede, The Netherlands, pp. 27–37.

Daelemans, W., A. van den Bosch, and J. Zavrel (1999). Forgetting exceptions is harmful in language learning. *Machine Learning 34*(1–3), 11–43.

Daelemans, W. M. P. and A. P. J. van den Bosch (1993). Tabtalk: Reusability in data-oriented grapheme-to-phoneme conversion. In *Proceedings of 3rd European Conference on Speech Communication and Technology (Eurospeech '93)*, Berlin, Germany, pp. 1459–1462.

Daelemans, W. M. P. and A. P. J. van den Bosch (1997). Language-independent data-oriented grapheme-to-phoneme conversion. In J. P. H. van Santen, R. W. Sproat, J. P. Olive, and J. Hirschberg (Eds.), *Progress in Speech Synthesis*, pp. 77–89. New York, NY: Springer.

Damper, R. I. and J. F. G. Eastmond (1997). Pronunciation by analogy: Impact of implementational choices on performance. *Language and Speech 40*(1), 1–23.

Damper, R. I. and Y. Marchand (2006). Information fusion approaches to the automatic pronunciation of print by analogy. *Information Fusion 7*(2), 207–220.

Damper, R. I., Y. Marchand, M. J. Adamson, and K. Gustafson (1999). Evaluating the pronunciation component of text-to-speech systems for English: A performance comparison of different approaches. *Computer Speech and Language 13*(2), 155–176.

Damper, R. I., Y. Marchand, J.-D. S. Marsters, and A. I. Bazin (2004). Aligning letters and phonemes for speech synthesis. In *Proceedings of 5th International Speech Communication Association (ISCA) Workshop on Speech Synthesis*, Pittsburgh, PA, pp. 209–214.

Damper, R. I., Y. Marchand, J.-D. S. Marsters, and A. I. Bazin (2005). Aligning text and phonemes for speech technology applications using an EM-like algorithm. *International Journal of Speech Technology 8*(2), 149–162.

Damper, R. I. and T. Soonklang (2007). Subjective evaluation of techniques for proper name pronunciation. *IEEE Transactions on Audio, Speech and Language Processing 15*(8), 2213–2221.

Damper, R. I., C. Z. Stanbridge, and Y. Marchand (2001). A pronunciation-by-analogy module for the Festival text-to-speech synthesiser. In *Proceedings of 4th International Speech Communication Association (ISCA) Workshop on Speech Synthesis*, Pitlochry, Scotland, pp. 97–102.

Davis, C. (2005). Shallow vs non-shallow orthographies and learning to read : An OECD workshop. Summary report, Cambridge University, Cambridge, UK.

Dedina, M. J. and H. C. Nusbaum (1991). PRONOUNCE: A program for pronunciation by analogy. *Computer Speech and Language 5*, 55–64.

Deshmukh, N., A. Le, J. Ngan, J. Hamaker, and J. Picone (1997). An advanced system to generate multiple pronunciations of proper names. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)*, Munich, Germany, pp. 1467–1470.

Dietterich, T. G., H. Hild, and G. Bakiri (1995). A comparison of ID3 and backpropagation for English text-to-speech mapping. *Machine Learning 18*(1), 51–80.

Elovitz, H. S., R. Johnson, A. McHugh, and J. E. Shore (1976). Letter-to-sound rules for automatic translation of English text to phonetics. *IEEE Transactions on Acoustics, Speech, and Signal Processing 24*(6), 446–459.

Fackrell, J. and W. Skut (2004). Improving pronunciation dictionary coverage of names by modelling spelling variation. In *Proceedings of 5th International Speech Communication Association (ISCA) Workshop on Speech Synthesis*, Pittsburgh, PA, pp. 121–126.

Fitt, S. (2000). Documentation and User Guide to UNISYN Lexicon and Post-Lexical Rules. Technical report, Centre for Speech Technology Research, Edinburgh, UK.

Font Llitjós, A. (2001). Improving pronunciation accuracy of proper names with language origin classes. Masters thesis, Carnegie Mellon University, Pittsburgh, PA.

Font Llitjós, A. and A. W. Black (2001). Knowledge of language origin improves pronunciation accuracy of proper names. In *Proceedings of 7th European Speech Communication Association International Conference (Eurospeech 2001)*, Aalborg, Denmark, pp. 1919–1922.

Font Llitjós, A. and A. W. Black (2002). Evaluation and collection of proper name pronunciations online. In *Proceedings of 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, Las Palmas, Spain, pp. 247–254.

Glushko, J. R. (1979). The organization and activation of orthographic knowledge

in reading aloud. *Journal of Experimental Psychology: Human Perception and Performance 5*(4), 674–691.

Golding, A. R. and P. S. Rosenbloom (1993). A comparison of Anapron with seven other name-pronunciation systems. *Journal of the American Voice Input/Output Society 14*, 1–21. Number unknown.

Hallahan, W. I. (1996). DECtalk software: Text-to-speech technology and implementation. *Digital Technical Journal 7*(4), 5–19.

Hensen, A. V. (1994). A self-organizing neural network (SONN). In *Proceedings of the Second ONOMASTICA Research Colloquium*, London, UK. Page numbers unknown.

Hieronymus, J. L. (1994). ASCII phonetic symbols for the world's languages: Worldbet. Technical memorandum, Bell Laboratories, Lucent Technologies, Murray Hill, NJ.

Hochberg, J., S. M. Mniszewski, T. Calleja, and G. J. Papcun (1991). A default hierarchy for pronouncing English. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*(9), 957–964.

International Phonetic Association (1999). *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet.* Cambridge, UK: Cambridge University Press.

ITU-T Rec.830 (Feb. 1996). *Subjective performance assessment of telephone-band and wideband digital codecs.* Geneva, Switzerland: International Telecommunications Union.

Katz, L. and L. B. Feldman (1981). Linguistic coding in word recognition: comparisons between a deep and a shallow orthography. In A. M. Lesgold and C. A. Perfetti (Eds.), *Interactive Processes in Readings*, pp. 85–106. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kienappel, A. K. and R. Kneser (2001). Designing very compact decision trees for grapheme-to-phoneme transcription. In *Proceedings of Eurospeech 2001*, Aalborg, Denmark, pp. 1–4.

Kiraz, G. A. and B. Möbius (1998). Multilingual syllabification using weighted finite state transducers. In *Proceedings of 3rd European Speech Communication Association (ESCA)/COCOSDA International Workshop on Speech Synthesis*, Jenolan Caves, Australia, pp. 71–76.

Klatt, D. and D. Shipman (1982). Letter-to-phoneme rules: A semi-automatic discovery procedure. *Journal of the Acoustical Society of America 72*(S1), S48.

Klatt, D. H. (1987). Review of text-to-speech conversion for English. *Journal of the Acoustical Society of America 82*(3), 737–793.

Lee, F. F. (1969). Reading machine: From text to speech. *IEEE Transactions on Audio and Electroacoustics 17*(4), 275–282.

Liberman, I., A. Liberman, I. Mattingly, and D. Shankweiler (1980). Orthography and the beginning reader. In J. Kavanagh and R. Venezky (Eds.), *Orthography, Reading, and Dyslexia*, pp. 137–153. Baltimore, OH: University Park Press.

Lucas, S. M. and R. I. Damper (1992). Syntactic neural networks for bi-directional text phonetics translation. In G. Bailly and C. Benoit (Eds.), *Talking Machines, Theories, Models and Designs*, pp. 127–141. North-Holland: Elsevier.

Lucassen, J. M. and R. L. Mercer (1984). An information theoretic approach to the automatic determination of phoneme base forms. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, San Diego, CA, pp. 42.5.1–42.5.4.

Marchand, Y. and R. I. Damper (2000). A multi-strategy approach to improving pronunciation by analogy. *Computational Linguistics 26*(2), 195–219.

Marchand, Y. and R. I. Damper (2007). Can syllabification improve pronunciation by analogy? *Natural Language Engineering 13*(1), 1–24.

Matsuoka, T., K. Ohtsuki, T. Mori, S. Furui, and K. Shirai (1996). Japanese large-vocabulary continuous-speech recognition using a business-newspaper corpus. In *Proceeding of 4th International Conference on Spoken Language Processing (ICSLP '96)*, Philadelphia, PA, pp. 22–25.

McCulloch, N., M. Bedworth, and J. Bridle (1987). NETspeak – A re-implementation of NETtalk. *Computer Speech and Language 2*(3), 289–301.

Möbius, B. (2003). Rare events and closed domains: Two delicate concepts in speech synthesis. *International Journal of Speech Technology 6*(1), 57–71.

Muller, J., H. Stahl, and M. Lang (1996). Predicting the out-of-vocabulary rate and the required vocabulary size for speech processing applications. In *Proceeding of 4th International Conference on Spoken Language Processing (ICSLP '96)*, Philadelphia, PA, pp. 1922–1925.

Müller, K. (2001). Automatic detection of syllable boundaries combining the advantages of treebank and bracketed corpora training. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France, pp. 402–409.

Oakey, S. and R. C. Cawthorn (1981). Inductive learning of pronunciation rules by hypothesis testing and correction. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, Vancouver, Canada, pp. 109–114.

Pagel, V. (2005). Private communication.

Quinlan, J. R. (1987). Learning decision trees. *Machine Learning 1*(1), 1–25.

Sampson, G. (1985). *Writing Systems*. London, UK: Hutchinson.

Schmid, H., B. Möbius, and J. Weidenkaff (2007). Tagging syllable boundaries with hidden markov models. In *Proceedings of Interspeech 2007*, Antwerp, Belgium.

Page numbers unknown.

Sejnowski, T. J. and C. R. Rosenberg (1987). Parallel networks that learn to pronounce English text. *Complex Systems 1*(1), 145–168.

Slowiaczek, L. (1990). Effects of lexical stress in auditory word recognition. *Language and Speech 33*(1), 47–68.

Spiegel, M. F. (1985). Pronouncing surnames automatically. In *Proceedings of the 1985 Conference of the American Voice Input/Output Society*, San Jose, CA, pp. 109–132.

Spiegel, M. F. (2003). Proper name pronunciation for speech technology applications. *International Journal of Speech Technology 6*(4), 419–427.

Sullivan, K. P. H. and R. I. Damper (1993). Novel-word pronunciation: A cross-language study. *Speech Communication 13*(3–4), 441–452.

Taylor, P. (2005). Hidden markov models for grapheme to phoneme conversion. In *Proceedings of Interspeech 2005*, Lisbon, Portugal, pp. 1973–1976.

The Onomastica Consortium (1995). The onomastica interlanguage pronunciation lexicon. In *Proceedings of 4th European Conference on Speech Communication and Technology (Eurospeech '95)*, Madrid, Spain, pp. 829–832.

Tian, J. (2004). Data-driven approaches for automatic detection of syllable boundaries. In *Proceedings of 8th International Conference on Spoken Language Processing (ICSLP '04)*, Jeju Islands, South Korea, pp. 61–64.

Torkkola, K. (1993). An efficient way to learn English grapheme-to-phoneme rules automatically. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Minneapolis, Minnesota, pp. II.199–II.202.

Trancoso, I. (1995). Issues in the pronunciation of proper names : The experience of the ONOMASTICA project. In *Proceedings in Workshop on Integration of Language and Speech*, Moscow, Russia. Page numbers unknown.

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

van Coile, B. (1991). Inductive learning of pronunciation rules with the Depes system. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Toronto, Canada, pp. 745–748.

van den Bosch, A., A. Content, W. Daelemans, and B. De Gelder (1994). Measuring the complexity of writing systems. *Journal of Quantitative Linguistics 1*(3), 178–188.

van den Bosch, A. and W. Daelemans (1993). Data-oriented methods for grapheme-to-phoneme conversion. In *Proceedings of 6th Conference of the European Chapter of the Association for Computational Linguistics*, Utrecht, The Netherlands, pp. 45–53.

Vitale, T. (1991). An algorithm for high accuracy name pronunciation by parametric speech synthesizer. *Computational Linguistics 17*(3), 257–276.

Weijters, A. (1991). A simple look-up procedure superior to NETtalk? In *Proceedings of International Conference on Artificial Neural Networks (ICANN '91)*, Espoo, Finland, pp. 1645–1648.

Yvon, F. (1994). Self-learning techniques for grapheme-to-phoneme conversion. In *Proceedings of the Second ONOMASTICA Research Colloquium*, London, UK. Page numbers unknown.

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