CHAPTER TWO

BACKGROUND

2.1 INTRODUCTION

This chapter provides background information with regard to the main topics discussed in subsequent chapters:

- Section 2.2 provides an overview of various approaches to pronunciation modelling;
- Section 2.3 describes the use of bootstrapping for the development of HLT resources in general; and
- Section 2.4 discusses current approaches to the creation of pronunciation dictionaries in a semi-automated fashion.

In this chapter, as in the remainder of this thesis, we use the ARPAbet symbol set (included in Appendix A) to demonstrate phonemic concepts.

2.2 PRONUNCIATION MODELLING

A pronunciation model for a specific language provides an accurate mechanism for letter-to-sound conversion, also referred to as grapheme-to-phoneme (g-to-p) conversion. Given the orthography of a word, grapheme-to-phoneme conversion provides a prediction of the phonemic realisation of that word. Where additional pronunciation characteristics such as stress or tone are predicted, this process is referred to as grapheme-to-phoneme conversion with stress and/or tone assignment. This can be the first of a two-phase process in pronunciation prediction: the first task being grapheme-to-phoneme conversion, the second phoneme-to-allophone conversion. The rules utilised in the latter phase are typically referred to as phonological rules, and are not always required explicitly, depending on the specific type of speech technology that will be utilising the dictionary. For example, a speech recognition system may either model phonological effects explicitly, or utilise a phonemic lexicon and rely on the context-dependent acoustic models to capture many of the phonological effects [26]. As the distinction between phonemes and phones is often blurred, we approach this differentiation in a pragmatic fashion in this thesis.

Pronunciations can be idiosyncratic, and not all pronunciation phenomena are regular to the extent of being predictable. Also, letter-to-sound conversion does not only depend on orthography: the phonemic outcome can (and does) depend on other linguistic features such as word part-of-speech, word morphology or word etymology. From a bootstrapping perspective, we are interested in approaches to the pronunciation prediction problem where additional linguistic resources are not available (or can be bootstrapped easily), and therefore we focus our attention on grapheme-to-phoneme conversion based mainly on orthography.

The remainder of this section provides an overview of current approaches to pronunciation modelling: Section 2.2.1 describes the manual development of pronunciation models, both the development of explicit pronunciation dictionaries and the handcrafting of grapheme-to-phoneme conversion rules, and Section 2.2.2 provides an overview of different approaches to the data-driven extraction of grapheme-to-phoneme conversion rules. As many of the data-driven approaches require grapheme-to-phoneme alignment prior to grapheme-to-phoneme rule extraction, approaches to grapheme-to-phoneme alignment are discussed separately in Section 2.2.3. Section 2.2.4 discusses an alternative speech processing approach that circumvents the need for explicit pronunciation modelling.

2.2.1 MANUAL DEVELOPMENT OF PRONUNCIATION MODELS

2.2.1.1 PRONUNCIATION DICTIONARIES

Many electronic pronunciation dictionaries (such as NETtalk [20] or OALD [18]) were created as digital versions of similar printed dictionaries. Classical printed pronunciation dictionaries typically only list word base forms, and for each word base form its ‘standard’ pronunciation. Pronunciation variants are only included when more than one distinct pronunciation exists for a single word (e.g. the past tense and present tense variants of the English word ‘read’: *r iy d* and *r eh d*). Electronic dictionaries that are frequently utilised in speech applications (such as CMUdict [17]) soon grow to include additional word forms (plurals and other derivatives), and multiple pronunciation variants, as required by the applications utilising the dictionary. Pronunciation variants can be generated automatically using phonological rule sets[1] or added according to a manual process.

Task-designed electronic pronunciation dictionaries, such as *FONILEX*, developed by Mertens and Vercammen [19], include systematic mechanisms to derive word variants from base forms. *FONILEX* specifically is a full-form lexicon (it lists the various word base forms separately) and

[1]The automatic extraction of phonological rules utilise techniques similar to those applied during grapheme-to-phoneme rule extraction, as described in Section 2.2.2.

provides an ‘abstract’ representation of each word, as well as three ‘concrete’ pronunciations representing three different speaking styles. The concrete pronunciations are derived automatically from the abstract representation via a set of phonological rewrite rules. In this way, regular variants are captured via phonological rules, rather than additional dictionary entries. Irregular variants are included as additional dictionary entries. A related approach, followed independently by Allen *et al* [27] and Coker *et al* [28], utilises morphemes as the stored unit, and obtains dictionary entries by combining these morphemes using a set of morphological rules. Here morphological rules are used to generate the word base form itself, which is not stored individually. It is interesting to note that *FONILEX* was compiled semi-automatically using grapheme-to-phoneme conversion, and verified manually – an approach that is related to the bootstrapping process investigated in this thesis.

2.2.1.2 PRONUNCIATION RULES

Manual pronunciation rules are typically developed according to the two-stage process described in Section 2.2; that is, two rule sets are created: one set of grapheme-to-phoneme rules, and a second set of phonological rules that generate the appropriate allophone (or allophones) per phoneme. Both rule sets are often augmented by a set of exceptions. These rule sets can be described according to different formalisms, a general formalism for a multi-level rewrite rule being:

$$\{a\}^* g \{b\}^* \rightarrow \{c\}^* p \{d\}^* \tag{2.1}$$

which, more typically, is simplified as: $\{a\}^* g \{b\}^* \rightarrow p$, where g indicates the grapheme being considered and p the specific phonemic realisation of g. $\{a\}^*$ and $\{b\}^*$ represent zero or more contextual elements to the left and the right of the grapheme (respectively) of words that this rule can be applied to, and $\{c\}^*$ and $\{d\}^*$ indicate how the word is amended (or not) during the application of this specific rule. Depending on the exact formalism, the left and/or right contexts of the left-hand side can either consist of graphemes only, or a combination of graphemes and phonemes, and similarly, the right-hand side can either be defined in terms of phonemes only, or a combination of graphemes and phonemes. A null (or empty) phoneme or grapheme may be utilised explicitly within the formalism. Furthermore, a single contextual element can also be used to represent a class of such graphemes or phonemes. Formalisms differ based on the order in which rules are applied, the direction in which rules are parsed, and whether a single rule or a sequence of all matching rules are applied when predicting a single word. Manually developed rewrite rules exist for a number of languages, including languages as diverse as English [29], Arabic [30] and isiXhosa [31].

Typically, the more modern the writing system of a language, the stronger the connection between the spoken and written form of a language, and the more regular the spelling system of the language[2]. Languages with a fairly recent spelling system (such as Swahili) have an almost direct correspondence between the orthography and the pronunciation of a word, while a language such as English or French

[2]As discussed further in Section 4.6.

includes significant historical 'baggage' in its spelling system. For languages with highly regular spelling systems, the manual development of a set of pronunciation rules can be a manageable task for a skilled linguist. For languages with less regular spelling systems this task becomes particularly arduous, with the set of words that that can be predicted correctly using the manually developed rule set only achieving larger sizes if amended by a sizeable exceptions dictionary. For example, the rule set developed by Elovitz *et al* [29], consisting of 329 rules for English, achieved only 25.7% word accuracy when evaluated by Damper *et al* [25] and 19.3% word accuracy when a modified version was evaluated by Bagshaw [32] (using different corpora). Semi-manually developed finite state transducer systems can achieve better performance [26], but require significant expertise to develop.

2.2.2 DATA-DRIVEN APPROACHES TO G-TO-P RULE EXTRACTION

Data-driven approaches to grapheme-to-phoneme rule extraction can be used to generalise from existing pronunciation dictionaries when handling out-of-vocabulary words in speech systems, and to compress information when requiring a pronunciation model in a memory-constrained environment. Such applications require a balance between the need for small rule sets, fast computation and optimal accuracy, and various approaches to pronunciation modelling have been defined to meet these requirements. Approaches include the application of neural networks [20, 33], decision trees [22–24, 34], Pronunciation by Analogy (PbA) models [32,35–38], instance-based learning algorithms such as Dynamically Expanding Context (DEC) [21,36] and IB1-IG [24], finite state transducers [39], Bayesian networks [40], and the combination of methods and additional information sources through meta-classifiers [41]. Many of these algorithms require grapheme-to-phoneme alignment prior to rule extraction, as discussed in Section 2.2.3.

Benchmarking these pronunciation prediction algorithms is difficult: There are few standardised pronunciation prediction tasks that are widely used, and the task itself is very sensitive to training/test set distributions. A strict evaluation of three of the data-driven approaches (a neural network, IB1-IG and PbA) can be found in [25]. Results obtained when applying different algorithms are discussed in further detail in Section 4.6.1; the remainder of this section provides an overview of the various approaches to grapheme-to-phoneme rule extraction mentioned above.

The automatic extraction of phonological rules utilise similiar techniques as those described here. Such rule sets are used to generate an allophonic representation for a phonemic pronunciation, as demonstrated by Ellison [42], Tajchman *et al* [43] and others, or to assign additional pronunciation characteristics such as stress to the pronunciation of the word [44]. The application of data driven techniques for the development of phonological rule sets is not discussed further: we rather focus our attention on the grapheme-to-phoneme conversion process specifically.

2.2.2.1 NEURAL NETWORKS AND DECISION TREES

A neural network was one of the first data-driven approaches to grapheme-to-phoneme rule set extraction demonstrated. A neural network was trained by Sejnowski and Rosenberg [20] using the

English NETtalk corpus, and later re-implemented by McCulloch and others as the NETspeak [33] system. Words were windowed with a fixed number of graphemes (between 3 and 11 graphemes) per window, and a feed-forward neural network was trained to associate each letter, surrounded by its graphemic window, with a specific phoneme outcome. A similar system was later evaluated by Damper *et al* [25].

Various decision tree based approaches have been demonstrated, including systems developed by Andersen *et al* [22, 45], Black *et al* [23] and Hakkinen *et al* [34], obtaining comparable results. The detail implementations differed based on various aspects, including the type of questions generated, the pruning method, the splitting criteria and detailed parameter choices. The algorithms were applied to different languages and corpora, and different evaluation processes used. Andersen *et al* compared a binary decision tree with Trie structures using both an English (NETtalk and CMUdict) and a French (ONOMASTICA) database [22]. Black *et al* utilised Classification and Regression Trees (CART) and English (OALD and CMUdict), French (BRULEX) and German (CELEX) dictionaries [23]. Hakkinen *et al* explicitly compared the performance of neural networks and decision trees for the English CMUdict task. Hakkinen *et al* found that neural networks provide better generalisation than decision trees when limited training data is available, and perform more consistently across mismatched test sets, while decision trees typically outperform neural networks where training and test data are closely matched [34].

2.2.2.2 PRONUNCIATION BY ANALOGY

Pronunciation by Analogy (PbA) models predict the pronunciation of a new word by searching through known words for matching sub-word parts. This set of algorithms was designed specifically for the task of grapheme-to-phoneme prediction. Originally suggested by Dedina and Nusbaum [46], the approach was further developed by Sullivan and Damper [35], Yvon [36, 47], Damper and Eastmond [37], Bagshaw [32], and Marchand and Damper [38].

Languages with irregular spelling systems such as English and French perform well within analogy-based frameworks, and for English, the best asymptotic results to date have been achieved with PbA [25]. Unfortunately, current versions of these algorithms can be ‘slow learners’, only approaching asymptotic accuracy for larger training dictionary sizes, as discussed further in Section 4.2. Depending on the amount of prior manipulation of the training data employed by PbA algorithms, these algorithms can be seen as a form of instance-based learning.

2.2.2.3 INSTANCE-BASED LEARNING

We use the term *instance-based learning* as used by Aha *et al* [48] to describe algorithms that generate classification predictions using specific instances from a set of training data, rather than using a generalised abstraction created from the training set, and do not differentiate among instance-based learning, memory-based learning or case-based reasoning. These algorithms all utilise ‘lazy learning’: rather than generalising from a training set, the entire training set is typically retained (in some form or another) and predictions are based on reasoning about these retained exemplars, analogous to the process of nearest neighbour classification[3].

In [24], Daelemans *et al* provide a strong argument for the utility of memory-based approaches for language processing tasks, noting that in many of these tasks exceptions tend to occur in 'groups or pockets in instance space'. As it is difficult to differentiate between actual noise inherent to language data[4] and small regular families of exceptions (that provide useful predictive information), Daelemans argues that exceptions should preferably be retained, as is inherent to standard instance-based learning. Two specific approaches that have been applied successfully to grapheme-to-phoneme conversion are (1) variations of *IB1-IG* [49], as developed and applied to the grapheme-to-phoneme task by Daelemans *et al* [24]; and (2) Kohonen's Dynamically Expanding Context (DEC) [50], initially applied by Torkkola [21] to the grapheme-to-phoneme task:

1. **IB1-IG**

   IB1-IG [24, 49] is in essence a k-nearest neighbour classifier that utilises as distance measure a weighted version of graphemic context overlap. Appropriate weighting of the graphemic context is an important aspect of the algorithm, and is attained through information gain techniques. Given a grapheme-to-phoneme aligned training dictionary, words are windowed, and a learning instance is generated per window (each instance focussing on a specific letter within the context of the rest of the window) and associated with a specific phonemic classification of that letter. Weights are associated with each feature based on a normalised measure of the amount of information the specific feature contributes to knowledge about the specific phonemic class (over the entire instance base). New words are predicted by finding the instances that are closest to the target word, using the weighted distance measure. Ties are resolved by considering frequency of outcome, and frequency of occurrence of the specific feature (where a feature defines both a letter and its position).

   Daelemans *et al* [24] evaluated this algorithm on the task of grapheme-to-phoneme conversion with stress assignment, using the CELEX database (as one of a set of language learning tasks considered), and found comparable accuracy rates between IB1-IG and a decision tree approach. The IB1-IG algorithm performed better than the C5.0 decision tree used for comparison: the difference in performance was slight (but significant) if the number of instances required for a decision tree node to be retained was chosen as 1 (similar to the IB1-IG approach); a larger number of required instances caused greater pruning of the decision tree, and decreased its performance. Damper *et al* [25] found that IB1-IG obtained higher accuracy than a neural network, but not the same level of asymptotic accuracy as PbA. Further results are provided by Hoste *et al* [41] in an evaluation of meta-classification techniques.

[3]It should be noted that the differentiation among techniques described in this section is not strict: for example, a decision tree learning algorithm that does not allow any pruning can also be seen as a form of instance-based learning.

[4]For example, as caused by true exceptions, or discrepancies in the way in which the lexicon was developed.

2. **DEC**

   Kohonen's Dynamically Expanding Context (DEC) [50], initially applied by Torkkola to the grapheme-to-phoneme problem [21], is another instance-based learning algorithm that predicts phoneme realisation based solely on graphemic context. In DEC, each rule specifies a mapping of a single grapheme to a single phoneme for a given left and right graphemic context, i.e is of the form: *(left-context,grapheme,right-context)* $\rightarrow$ *phoneme*.

   Rules are extracted by finding the smallest context that provides a unique mapping of grapheme to phoneme. If an $n-$letter context is not sufficient, the context is expanded to either the right or the left. This 'specificity order' influences the performance of the algorithm. The set of extracted rules are stored as a hierarchical tree, with more general rules at the root, and more specific rules at the leaves. The tree is traversed from the root to the leaves, and the rule at the first matching leaf (the rule describing the largest matching context) is used to predict the specific grapheme-to-phoneme realisation. If no leaf is matched, the most probable outcome of the last matching leaf is used, as can be estimated from the training data. If the extracted 'rule set' is allowed to contain contexts of an arbitrary size, no training words are discarded, and the tree structure is simply used to arrange the set of all training instances in an efficient structure.

2.2.2.4 ALTERNATIVE APPROACHES

A number of further approaches to pronuncation modelling exist, including:

1. Finite state transduction, as demonstrated by Luk and Damper [39], and more recently by Hazen *et al* [26]. Finite state transduction as used in [26] requires significant linguistic specification, while Luk and Damper's approach requires less linguistic input but makes a number of (restrictive) assumptions in order to create a trainable system.

2. The application of Bayesian networks for grapheme-to-phoneme conversion [40]. Bayesian networks are more typically used for pronunciation variation modelling, rather than phonemic base form generation.

3. The use of hierarchical systems of meta-classifiers, and even meta-meta-classifiers as investigated by Hoste *et al* [41].

Various of these approaches can be utilised during bootstrapping, as discussed further in Section 4.2.

2.2.3 GRAPHEME-TO-PHONEME ALIGNMENT

The majority of data-driven approaches to grapheme-to-phoneme rule extraction first require that the training dictionary be aligned on a grapheme-to-phoneme basis. For languages with alphabetic writing systems[5], each grapheme is mapped to its corresponding phoneme, and phonemic or graphemic

[5]For ideographic, pictographic, syllabic or even moraic languages, a more complex process is required – see for example [51] for a comparison of alignment approaches for Japanese.

nulls inserted where required: A phonemic null is inserted where a single phoneme is produced from more than one grapheme; a graphemic null where a single grapheme results in more than one phoneme. In languages where graphemic nulls are rare, graphemic exceptions that can map to more that one phoneme (such as $x \rightarrow k\ s$) can be replaced with two pseudo-graphemes (e.g. replacing x with $x\ X$) and only phonemic nulls inserted. This technique, suggested by Pagel *et al* [52], results in fewer alignment errors.

Initial data sets used for grapheme-to-phoneme benchmarking (such as *NETtalk* [20]) were hand aligned. Dalsgaard, Andersen and others [53, 54] applied forced Viterbi alignment [55] to create automatic grapheme-to-phoneme alignments, based on the probabilities *P(grapheme i | phoneme j)*. Initial probabilities were obtained from words and pronunciations that have equal length. This approach provides fairly accurate alignments: when benchmarked against the *NETtalk* hand alignments, Andersen *et al* achieved a word alignment accuracy of 83.7% and a phoneme alignment accuracy of 93.2% [22]. It should be noted that the *NETtalk* hand alignments may not be the ideal benchmark to use for measuring alignment accuracy, as discussed in more detail in Section 4.4. Black *et al* [23] used a similar alignment approach but defined a candidate set to restrict misalignments. In Black's approach the possible grapheme-to-phoneme mappings are specified prior to alignment, and used to restrict the alignment options during Viterbi alignment.

2.2.4 GRAPHEME-BASED SYSTEMS

The discussion up to this point has assumed that a pronunciation model is a required component for a variety of speech processing systems, including automatic speech recognition systems. Schillo *et al* [56] demonstrated an alternative approach by introducing the concept of grapheme-based speech recognition: rather than using a pronunciation dictionary, graphemes are used directly as basis for the acoustic sub-units modelled. This grapheme-based approach results in surprisingly accurate systems. Since the perplexity of the language model has a significant effect on the accuracy of the system, a strong language model compensates well for an inaccurate pronunciation model. The results obtained by Schillo *et al* were independently confirmed by Kanthak and Ney [57] and Killer [58]. While grapheme-based systems are conceptually less complete than system that incorporate an explicit pronunciation dictionary, grapheme-based systems for languages with fairly regular spelling systems (such as Italian, Spanish or Dutch) do not seem to be significantly less accurate than phoneme-based systems, especially in the presence of a strong language model, exhibiting a less than 2% relative decrease in accuracy in [57]. For languages with less regular systems, the decrease in accuracy becomes more noticeable: In [57] a 25.7% relative decrease in accuracy was observed for an English system with a word trigram perplexity of 124.5.

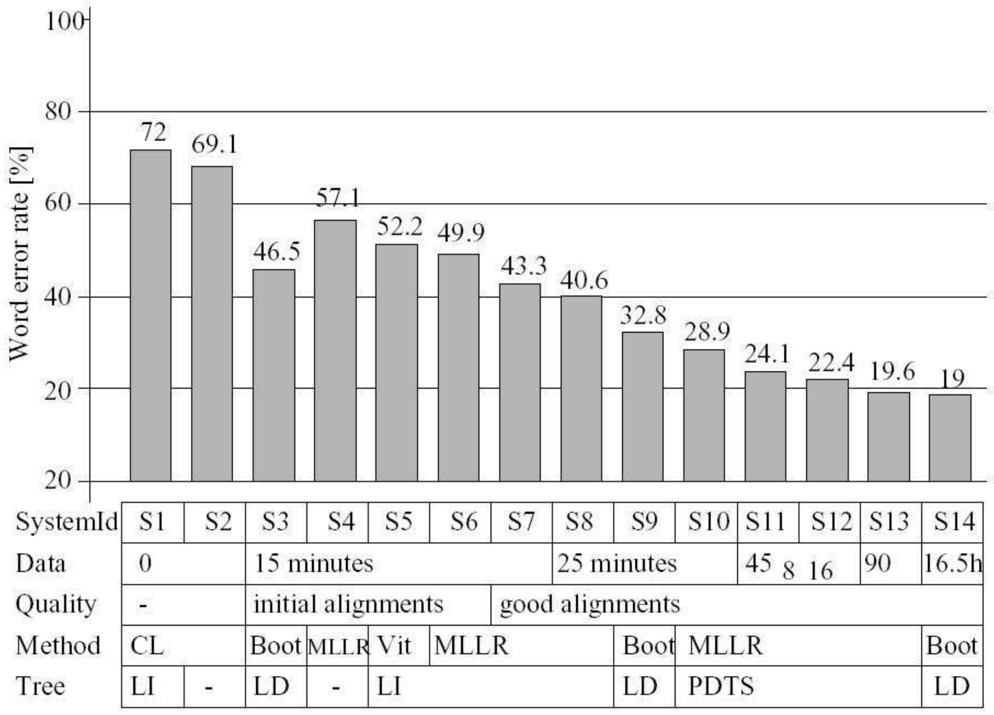
2.3 BOOTSTRAPPING OF HLT RESOURCES

We use the term 'bootstrapping' to describe an iterative process whereby a model of some form is improved via a controlled series of increments, at each stage utilising the previous model to generate the next one. This is a broader definition than often employed in machine learning, where bootstrapping typically indicates a semi-supervised approach to learning, where a small set of labelled instances is used to seed a classifier, label unclassified data, and retrain the classifier [59, 60]. Both the above interpretations should not be confused with the use of this term in the field of Statistics, where it can also indicate a statistical method for estimating the sampling distribution of an estimator by resampling with replacement from the original sample [61].

Bootstrapping can be a useful technique during language resource development, and has been used extensively in the creation of resources required by automatic speech recognition systems [3, 9–11]. In speech recognition, a bootstrapping technique is often combined with some form of cross-language information sharing. For example, when acoustic models are developed for a new target language, an automatic speech recognition system can be initialised with pre-developed models from an acoustically similar source language, and these initial models improved through an iterative process whereby audio data in the target language is automatically segmented using the current set of acoustic models, the models retrained and the target data re-segmented via a set of incremental updates.

The potential saving in resource requirements achieved through such a process was well demonstrated by Schultz and Waibel [12]. For example, in a set of experiments conducted on a Portuguese system, Schultz and Waibel obtained near-equal performance using either a fairly large amount (16.5 hours) of target data, or adapting multilingual models through a combination of bootstrapping and adaptation, using 90 minutes of target data. The increase in performance using different techniques is illustrated in Figure 2.1. Here, *Data* refers to the amount of target language data used and *Quality* refers to the quality of the alignments: *initial* alignments are generated by the multilingual system, while *good* alignments are updated based on improving systems. *Method* refers to the adaptation method used: using the unadapted initial system in a cross-language transfer approach (CL), Viterbi training using the alignments from the initial system (Vit), Maximum Likelihood Linear Regression adaptation of the initial system using the target data (MLLR), or bootstrapping (Boot). Bootstrapping consists of the following phases per bootstrapping cycle: creating initial alignments, Viterbi training, model clustering, retraining and writing improved alignments. *Tree* refers to the decision tree used for clustering: the original multi-lingual language independent tree (LI), a Portuguese language dependent tree (LD) or a tree built using the Polyphone Decision Tree Specialisation (PDTS) process[6] [12]. This example illustrates both the cross-language re-use of information – seeding the acoustic models using a related language – and the essence of a bootstrapping approach: iteratively improving acous-

[6]A standard context modelling technique is to cluster models using a CART-based clustering technique and a splitting criterion based on maximum entropy gain. The Polyphone Decision Tree Specialisation (PDTS) technique was proposed by Schultz as a mechanism to adapt the context modelling based on the target data, by restarting the decision tree growing process according to the target data available, resulting in significant improvements [62].



SystemId	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14
Data	0		15 minutes					25 minutes			45	8 16	90	16.5h
Quality	-		initial alignments				good alignments							
Method	CL		Boot	MLLR	Vit	MLLR			Boot	MLLR				Boot
Tree	LI	-	LD	-	LI				LD	PDTS				LD

Figure 2.1: *Experimental results when applying cross-language re-use of acoustic information techniques in the bootstrapping of a Portuguese system, from [12].*

tic models by utilising the models developed during the previous bootstrapping cycle to re-align the data, and retrain the models.

Additional language resource development tasks that have been shown to benefit from some form of bootstrapping include the development of parallel corpora [13], morphological dictionaries [14], text categorization [63], automatic audio alignments [64], grammar parsers [65], morphological analysers [15], linguistically tagged corpora [16], and the development of pronunciation lexicons, as discussed in Section 2.4.

2.4 THE AUTOMATED GENERATION OF PRONUNCIATION DICTIONARIES

In this section, we consider automated and semi-automated approaches to the generation of pronunciation dictionaries in a new language, referring to two types of approaches: Stuker [66] investigated ways in which existing phoneme recognisers can be used to generate a pronunciation dictionary for a new language, utilising audio data and word-level transcriptions in the target language. Using nine mono-lingual and a multi-lingual phoneme recogniser, phoneme recognition of the audio data is performed, and different voting and normalisation techniques are used to obtain a hypothesized pronunciation (or pronunciations) per words. This technique does not currently result in usable dictionaries, but further work is in progress.

A demonstrably successful approach to the semi-automated generation of pronunciation dictionaries, is the use of bootstrapping within the Festival Text-to-Speech System [67]. This system includes a rule extraction component based on Classification and Regression Trees, which can be used to generate letter-to-sound rules from a small lexicon. This lexicon is then grown iteratively by submitting additional words to the system, and having a human verify the correctness of the predictions. This process was recently demonstrated by Maskey *et al* [68], utilising an approach that is analogous to the approach used in this thesis. Maskey *et al* developed a Nepali pronunciation dictionary by iteratively extracting a grapheme-to-phoneme rule set, predicting a set of additional dictionary entries (varying from 100 words per cycle initially to 5000 words per cycle later in the process), identifying a subset of these words based on a calculated confidence score, and having these corrected by a Nepali speaker. In a related approach, the *FONILEX* dictionary was compiled semi-automatically using grapheme-to-phoneme conversion, and verified manually [19].

2.5 CONCLUSION

This chapter provided background on the pronunciation modelling task, and described various approaches to pronunciation modelling, focussing on data-driven techniques. The pronunciation modelling topic is addressed further in Chapter 4, where we define a grapheme-to-phoneme rule extraction mechanism suitable to bootstrapping. The current chapter also provided a brief overview of prior work related to the bootstrapping of HLT resources; this discussion continues in Chapter 3 with the definition of a general model for the bootstrapping of HLT resources.