World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:2, 2008

Inferring Hierarchical Pronunciation Rules from a Phonetic Dictionary

Erika Pigliapoco, Valerio Freschi, and Alessandro Bogliolo

Abstract**—**This work presents a new phonetic transcription system based on a tree of hierarchical pronunciation rules expressed as context-specific grapheme-phoneme correspondences. The tree is automatically inferred from a phonetic dictionary by incrementally analyzing deeper context levels, eventually representing a minimum set of exhaustive rules that pronounce without errors all the words in the training dictionary and that can be applied to out-of-vocabulary words.

The proposed approach improves upon existing rule-tree-based techniques in that it makes use of graphemes, rather than letters, as elementary orthographic units. A new linear algorithm for the segmentation of a word in graphemes is introduced to enable out-of-vocabulary grapheme-based phonetic transcription.

Exhaustive rule trees provide a canonical representation of the pronunciation rules of a language that can be used not only to pronounce out-of-vocabulary words, but also to analyze and compare the pronunciation rules inferred from different dictionaries. The proposed approach has been implemented in C and tested on Oxford British English and Basic English. Experimental results show that grapheme-based rule trees represent phonetically sound rules and provide better performance than letter-based rule trees.

Keywords**—**Automatic phonetic transcription; pronunciation rules; hierarchical tree inference.

I. Introduction

AUTOMATIC text pronunciation is an important processing task in computational linguistics. A lot of work has been done in the last decade in the field of letter-to-sound transformations. In particular, the development of *text-to-speech* (TTS) systems has driven the research in automatic phonetic transcription. In fact, processing modules for aligning written letters to corresponding *phonemes* (i.e., meaningful sounds) are always among the first tasks in the work-flow of TTS systems [15], [9].

Current approaches to automatic phonetic transcription of written text can be mainly classified in *rule-based* and *data-driven* approaches [5], [6], [8].

Rule-based approaches (that were the first to be developed) follow the methodological lines of generative grammars rooted in computational linguistics [3], formalizing pronunciation production rules of the type $A\langle B\rangle C \rightarrow P$, whose semantic is: substring B is mapped to phoneme P when its left context is A and its right context is C. From a computational point of view, rules are taken in input and used by the TTS system to derive the phonetic transcription of arbitrary input texts. Needless to say, the accuracy of rule-based approaches strongly depends on the quality of the rules. Deriving meaningful rules that capture the complex relationship between written text and pronunciation is a challenging task by itself (that also involves theoretical issues) especially in languages rich in spelling irregularities and exceptions, like English. To handle, or overcome, this problem different methods have been devised, mainly falling in the category of data-driven approaches.

Data-driven methods try to exploit the knowledge which is implicit in pronunciation databases, such as dictionaries with phonetic transcriptions. The wider the size of the dictionary, the more effective the method. The main challenge here is to infer the phonetic transcription of words that are not in the dictionary. This issue is known in literature as the *out-of-vocabulary* (OOV) problem.

A possible solution to OOV pronunciation relies on substring similarities: the OOV word is split into substrings that are matched to substrings belonging to the dictionary (i.e., with known pronunciation). The result of this matching process is a *pronunciation lattice*, i.e., a graph data structure used to represent the matching of the input string with the dictionary (sub)strings. In fact, the nodes of the graph represent letters $W(i)$ of the input string W and are labeled by phonemes associated with them, while an arc between node i and node j represents a matching (sub)string that starts with $W(i)$ and ends with $W(j)$. Each arc is also labeled with the phonetic transcription of the substring it represents, and with a frequency count. Paths from the beginning to the end of the lattice are scored (e.g., by the sum of frequencies of each arc) and best paths are taken as candidates for pronunciation. This type of algorithmic technique (called *pronunciation by analogy* by Dedina and Nusbaum [7]) is basically heuristic: Different ways to score the paths can lead to different paths. A multi-strategy approach that takes into account different possible scoring systems has been proposed to increase efficiency in the decision process [5].

Another technique for automatic phonetic transcription applied to the OOV problem entails constructing the orthographic neighborhood of the given OOV word (i.e., the set of in-vocabulary words sufficiently close to it according to a given distance metric) and deriving the pronunciation for the OOV word from an alignment procedure of phonetic transcriptions of the orthographic neighborhood [2].

A different kind of data-driven solutions has been recently proposed that makes use of *hidden Markov models* (HMM) to derive a grapheme-to-phoneme conversion [16]. In the HMM approach, phonemes represent hidden states, transitions between hidden states represent the probability that a phoneme is followed by another one, and graphemes are derived as observations, according to the HMM generative model and to the probability distributions associated to each state (that

Authors are with ISTI - University of Urbino, Urbino, Italy (e-mails: (erika.pigliapoco,valerio.freschi,alessandro.bogliolo)@uniurb.it).

have to be learned during the training phase).

A hybrid approach between data-driven and rule-base techniques consists of automatically extracting a set of rules from a phonetic dictionary, taking into account left and right contexts in order to reduce pronunciation ambiguity [4], [10], [11], [13], [14], [17]. In particular, the algorithm proposed by Hochberg *et al.* [10] is based on the alignment between characters and phonemes and it proceeds by inducing a hierarchical tree of rules that become progressively more specific at deeper levels of the tree as larger contexts are considered. Starting from level 1, that provides a context-free pronunciation rule for the given letter, the algorithm proceeds by choosing a context-letter at a time and by adding child nodes to represent context-specific exceptions to the context-free rule. At each step, the most significant incremental context is chosen. Daelemans *et al.* [4] developed a similar algorithm that constructs a rule tree by enforcing a fixed criterion (based on static information-theoretical analysis) for choosing context letters.

This paper proposes a similar approach that infers a tree of hierarchical pronunciation rules from a phonetic dictionary, but it uses graphemes in place of letters. The result is a set of grapheme-to-phoneme (rather than letter-to-sound) associations that are directly comparable with those used in generative grammars. Moreover, the rule tree provides a canonical representation that can be used to analyze the empirical pronunciation rules (and exceptions) of a given language and to compare the rules of different languages, as exemplified in Section III-F. The depth of the tree can be limited to obtain different tradeoffs between size and accuracy. If no limits are imposed, the tree represents exhaustive pronunciation rules that read without errors all the words in the training dictionary.

The proposed approach has been implemented in C and compared with the algorithm by Daelemans [4] in terms of accuracy, efficiency, and phonetic soundness of the inferred rules. Finally, the algorithm has been applied to infer and compare the empirical pronunciation rules of British English and Basic English [12].

II. Automatic Text Pronunciation

Before outlining the proposed approach to automatic text pronunciation, operative definitions need to be introduced of the terms 'phoneme', 'grapheme' and 'phonogram', that are extensively used hereafter.

A *phoneme* can be defined as the smallest phonetic unit capable of causing a distinction in meaning. All the phonemes of a given language can be associated with a subset of the *International Phonetic Alphabet* (IPA)[1]. IPA symbols are then used to denote the corresponding phonemes. When ASCII characters are used in place of the standard IPA notation, phonemes are represented within slashes (e.g., /@/).

A *grapheme* is a sequence of one or more characters that may be pronounced as a single phoneme. Graphemes are usually represented within angle brackets. For instance, according to our definition, ⟨kn⟩ is an English grapheme, since in many words it is pronounced as a single phoneme (/n/ as in 'known'). However, ⟨k⟩ and ⟨n⟩ are also English graphemes by themselves, since they occur separately in many words. Moreover, even if they appear together, they may be treated as separate graphemes as in the composite word 'banknote', where 'kn' has to be regarded as ⟨k⟩⟨n⟩ and pronounced /k//n/.

A *phonogram* is a grapheme-phoneme pair where the grapheme is one of the possible orthographic representations of the phoneme and the phoneme is one of the possible pronunciations of the grapheme. Since in many languages there is no 1-1 correspondence between graphemes and phonemes, there are many more phonograms than graphemes and phonemes.

The proposed algorithm starts from: i) the set of the ASCII representations of the phonemes of the target language, ii) the set of the graphemes of the target language, iii) a conventional table of phonograms and iv) a consistent *phonetic dictionary*, i.e., a list of words with corresponding phonetic transcriptions.

The phonetic dictionary is parsed and used as a training set to build a tree of statistical pronunciation rules that can be eventually applied to OOV words.

A. Theoretical Foundation

According to our definition of grapheme, the segmentation of a word in graphemes is induced by its pronunciation through the phonogram table. For instance, given the English word 'known' and its phonetic transcription /n//@U//n/, the phonogram table suggests that the word is composed of the graphemes ⟨kn⟩⟨ow⟩⟨n⟩ that are aligned to the omologous phonemes. Similarly, given the English word 'banknote' and its phonetic transcription /b//&//N//k//n//@U//t/, the corresponding segmentation is ⟨b⟩⟨a⟩⟨n⟩⟨k⟩⟨n⟩⟨o⟩⟨t⟩⟨e⟩. These two examples point out that graphemes are not uniquely determined by the orthography (for instance, 'kn' is split into ⟨k⟩⟨n⟩ in 'banknote' and taken as a single grapheme ⟨kn⟩ in 'known') and that words may contain silent graphemes (such as ⟨e⟩ in 'banknote'). Given a word, its phonetic transcription and a phonogram table, its segmentation in graphemes can be determined by means of a branch and bound decision process.

Since the aim of this work is building a tree of pronunciation rules to be possibly applied to OOV words, it is impossible to rely on the knowledge of the phonetic transcription of a word to obtain its segmentation in graphemes. Rather, it is necessary to obtain first the graphemes from the orthography, and then the phonetic transcription from the sequence of graphemes. This can be done without ambiguity by extending the set of graphemes as outlined in the following.

A grapheme x *contains* a grapheme y if y is a substring (i.e., a subset of contiguous characters) of x. In this case, y is said to be a *sub-grapheme* of x. The sequence of graphemes $\mathcal{P} = (y_1, y_2, ..., y_n)$ is a *partition* of x if and only if x can be obtained from the concatenation of y_1, y_2, ..., y_n.

Two graphemes x and y are *overlapping* if the tail (i.e., the last part) of x is equal to the head (i.e., the first part) of y. For instance ⟨kn⟩ contains ⟨k⟩ and ⟨n⟩, $\mathcal{P}$=(⟨k⟩,⟨n⟩) is a partition of ⟨kn⟩, while ⟨ss⟩ and ⟨sh⟩ are overlapping graphemes.

G is a *complete grapheme set* if and only if, for each pair of overlapping graphemes x and y in G, there is a grapheme z in G that contains both x and y. For instance, if G contains ⟨ss⟩ and ⟨sh⟩, it is not complete unless it also contains ⟨ssh⟩.

[1] International Phonetic Association, http://www2.arts.gla.ac.uk/IPA/index.html, 1996

P is a *complete phonogram table* if and only if the phonetic transcriptions associated with each grapheme include all combinations of the phonemes associated with the graphemes of its possible partitions. In our example, a complete phonogram table associates to grapheme ⟨kn⟩ not only its single-phoneme prononciation /n/, but also the concatenation of the possible phonemes associated with ⟨k⟩ and ⟨n⟩, such as /k//n/. Hereafter, the concatenation of the phonemes associated with a single grapheme is represented by a dot, and the resulting sequence of phonemes is treated as a single phoneme (/k.n/).

A partition $\mathcal{P}$ of a given word w, defined over grapheme set G, is a said to be a *minimum partition* for w if it contains the minimum number of graphemes.

Theorem 1. Given a word w and a complete grapheme set G, the minimum partition $\mathcal{P}$ of w defined over G is unique and it can be incrementally constructed in linear time during the parsing of w.

Proof. The uniqueness of the minimum partition $\mathcal{P}$ can be demonstrated by assuming, by contradiction, that there are two different minimum partitions (namely, $\mathcal{P}1$ and $\mathcal{P}2$) of the same word w. In order for $\mathcal{P}1$ to be different from $\mathcal{P}2$ there must be at least a character c of w that is considered to be part of different graphemes in $\mathcal{P}1$ and $\mathcal{P}2$ (namely, $g1$ and $g2$). Since the two graphemes cover the same character c of w they must be either contained in each other or overlapping. If $g1$ is contained in $g2$ (or viceversa) then $\mathcal{P}1$ (or $\mathcal{P}2$) is not a minimum partition, that's a contradiction. If $g1$ and $g2$ are overlapping there must be in G a grapheme that contains both of them, so that neither $\mathcal{P}1$ nor $\mathcal{P}2$ are minimum partitions, leading ot a contradiction.
The minimum partition can be determined in linear time by using a *keyword tree* [1] to represent all the graphemes in G: Edges are associated with letters; existing graphemes (keywords) are associated with paths from the root; graphemes with the same prefix share the first part of their paths. The letters of the word are read one at the time, following the corresponding path on the tree. When the path cannot be further extended the grapheme corresponding with the path is added to $\mathcal{P}$ and the traversal is restarted from the root of the tree.

Theorem 2. Given a word w, a complete grapheme set G, a complete phonogram table P and a minimum partition $\mathcal{P}$ of w, the correct pronunciation of w is one of the possible configurations of the phonemes associated with the graphemes of $\mathcal{P}$ according to P.

Proof. If g is a grapheme in $\mathcal{P}$, then there are no larger graphemes covering the same letters of w or part of them, or otherwise $\mathcal{P}$ would not be a minimum partition. By construction, the extended phonogram table P contains all possible pronunciations of g, including its correct pronunciation in the context of w. Since this is true for all the graphemes in $\mathcal{P}$, the correct pronunciation of w is one of the configurations of the phonemes associated with the graphemes of $\mathcal{P}$ according to P.

Theorems 1 and 2 provide the theoretical basis for our approach. In fact, they allow us to: i) split a word in graphemes independently of its pronunciation, ii) express pronunciation rules in terms of context-specific grapheme to phoneme associations and iii) obtain the pronunciation of a word by applying the pronunciation rules to the sequence of its graphemes.

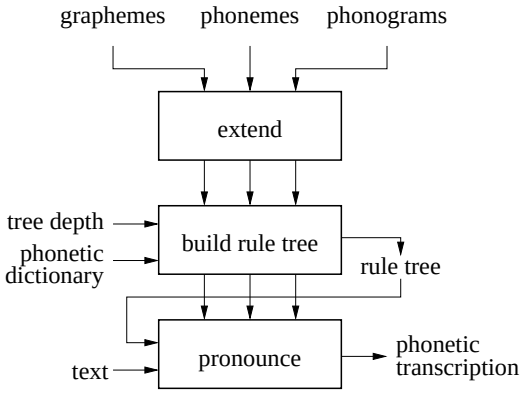

Fig. 1. Tool flow of the proposed approach.

It is worth noting that the complete grapheme set, the complete phonogram table and the corresponding phoneme set are automatically generated in a pre-processing step by extending the grapheme set, the phonogram table and the phoneme set provided by the user. The usage of the extended data structures in place of the original ones is needed to meet the completeness requirements of Theorems 1 and 2. However, extended data structures are kept transparent to the user and the phonetic transcription provided by the algorithm is coherent with the original grapheme set and phonogram table. The proposed tool flow is schematically represented in Figure 1.

B. Rule Tree Structure

Pronunciation rules are represented by means of a tree whose nodes are associated with the graphemes to be pronounced and with their contexts. The complete context of a grapheme is the word it belongs to. A phonetic dictionary uses the entire context to pronounce each grapheme. However, once a word is partitioned in graphemes, the context of each grapheme can be incrementally specified by looking at the surrounding graphemes one at the time, according to a language-specific priority. Information theoretical analysis of many languages demonstrated that the contribution of a context grapheme to the disambiguation of the phonetic transcription decreases with its distance from the grapheme to be pronounced. Moreover, right-context graphemes are more important than left-context graphemes at the same distance [4]. Hence, context graphemes are alternatively taken from the list of graphemes that follow (right-context) and precede (left-context) the current one in the partition of the word. The context of ⟨**kn**⟩ in 'banknote' would then be incrementally expressed as: ⟨**kn**⟩⟨o⟩ (level 1), ⟨n⟩⟨**kn**⟩⟨o⟩ (level 2), ⟨n⟩⟨**kn**⟩⟨o⟩⟨t⟩ (level 3), eventually getting to the entire word ⟨b⟩⟨a⟩⟨n⟩⟨**kn**⟩⟨o⟩⟨t⟩⟨e⟩ (level 6).

The corresponding tree is partially represented in Figure 2. At each node of the tree, a statistical pronunciation rule is provided based on the partial context information available at

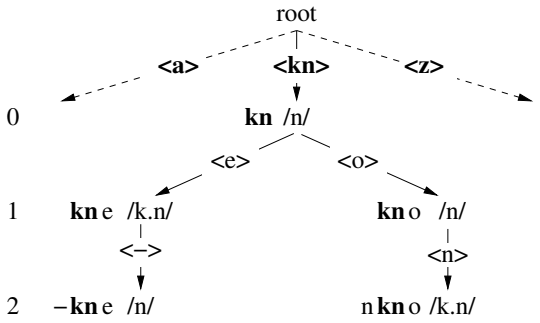

Fig. 2. Tree-structured representation of the pronunciation rules of English grapheme ⟨kn⟩.

that level. For instance, in English the most likely pronunciation of ⟨kn⟩ is /n/ at level 0, but it may be /k.n/ at level 1 (if the right context is ⟨e⟩) and at level 2 (if the right context is ⟨o⟩ and the left context is ⟨n⟩). It is worth noting that an empty grapheme ⟨-⟩ is used to represent word boundaries in the context tree. For instance, the left-hand side leaf node of the rule tree of Figure 2 represents the pronunciation of ⟨kn⟩ when it appears at a beginning of a word (i.e., when its left context is an empty grapheme) and its right context is ⟨e⟩, like in 'knelt'.

The rule tree is *irredundant*, in that it contains a node if and only if it is strictly required to represent a context-dependent exception to the pronunciation rule expressed by its parent node, and *canonical*, in that it is uniquely determined by the rules it represents.

For instance, level-1 node '**kn**e' of figure 2 represents an exception to the pronunciation rule of its parent node. Its level-2 child ('-**kn**e') represents a further exception, even if it provides the same pronunciation of the level-0 node. On the right-hand side of the tree, level-1 node '**kn**o' is associated with the same phoneme of its level-0 parent. Hence, it doesn't represent an exception by itself, but it is strictly required anyway to lead to a level-2 exception.

The sub-tree reported in figure 2 represents the following hierarchical pronunciation rules:

grapheme and context	pronunciation
⟨kn⟩	/n/
⟨kn⟩e	/k.n/
-⟨kn⟩e	/n/
n⟨kn⟩o	/k.n/

It is worth noting that there is no rule associated with node '**kn**o', since it doesn't represent an exception, as discussed above.

C. Rule-Tree Construction and Characterization

The rule tree is incrementally constructed and characterized, level by level, on the basis of the training phonetic dictionary. At each iteration the entire dictionary is parsed, each word is segmented into maximum-size graphemes and the graphemes are aligned with the phonemes in the corresponding phonetic transcription. The alignment is determined by means of a branch-and-bound procedure driven by the complete phonogram table. Context-specific grapheme-phoneme associations are deduced from the alignment and used first to update the structure of the tree, and then to update the leaf-node statistics. Reminding that each node represents a grapheme in a partially-specified context, node statistics express the context-dependent probability distribution of the phonemes. A grapheme g that appears in a given word of the training dictionary aligned with phoneme p contributes to node statistics by incrementing the counter associated with phoneme p in the node corresponding to g and to its partial context (that is found by traversing the rule tree from its root, taking at level 0 the branch associated with g, and at subsequent levels the branches associated with its context graphemes). A node counter is also incremented to be used to compute phoneme frequencies from phoneme counts.

The tree construction algorithm can be outlined as follows:

1) Create a root node with a null grapheme associated with a silent phoneme.
2) Add level-0 child nodes associated with all the graphemes in G.
3) Parse the dictionary and update level-0 (context-independent) statistics.
4) Associate the most-likely pronunciation to level-0 nodes.
5) For k from 1 to tree_depth
   a) Parse the dictionary and add level-k nodes (associated with level-k context graphemes) whenever the context-specific pronunciation is different from the most likely level-(k-1) pronunciation.
   b) Parse the dictionary and update level-k node statistics
   c) Associate the most-likely pronunciation to level-k nodes.

D. Rule-Tree Application

Once the rule tree has been inferred from a phonetic dictionary, it can be applied to pronounce any word (say, w) according to the following procedure:

1) find a minimum partition $\mathcal{P}$ of w
2) for each grapheme g in $\mathcal{P}$
   a) traverse the rule tree from the root, following the path corresponding to g and to its incremental context, until a node n is reached that has no children associated with the next context grapheme
   b) use the phoneme associated with node n to pronounce g
3) return the phonetic transcription of w splitting composite phonemes into separate IPA phonemes

Consider, for instance, English word w='known'. Its minimum partition is:

$$\mathcal{P}=(\langle kn\rangle,\langle ow\rangle,\langle n\rangle).$$

Pronouncing the first grapheme, ⟨kn⟩, entails following the path associated with its incremental context, that is: root→⟨kn⟩→⟨ow⟩→⟨-⟩→⟨n⟩. The traversal of the rule tree, however, stops at level-0 node '**kn**', since it has no children (i.e., no exceptions) associated with context grapheme ⟨ow⟩. Hence, the pronounciation of ⟨kn⟩ is /n/.

The pronunciation of ⟨kn⟩ in 'banknote' can be found by following the path:

$$\text{root}\rightarrow\langle\text{kn}\rangle\rightarrow\langle\text{o}\rangle\rightarrow\langle\text{n}\rangle\rightarrow\langle\text{t}\rangle\rightarrow\langle\text{a}\rangle\rightarrow\langle\text{e}\rangle\rightarrow\langle\text{b}\rangle$$

that stops at node 'n**kn**o', associated with phoneme /k.n/.

Finally, the context path for grapheme ⟨kn⟩ in 'knelt' is:

$$\text{root}\rightarrow\langle\text{kn}\rangle\rightarrow\langle\text{e}\rangle\rightarrow\langle\text{-}\rangle\rightarrow\langle\text{l}\rangle\rightarrow\langle\text{-}\rangle\rightarrow\langle\text{t}\rangle$$

that stops at '-**kn**e', associated with phoneme /n/.

III. Experimental Results

The proposed methodology (hereafter called *Grapheme-based Rule Tree*, *GRT*) was implemented in C and applied to *British English* and *Basic English* [12]. The approach by Daelemans *et al.* [4] (hereafter denoted by *LRT*, that stays for *Letter-based Rule Tree*) was also implemented and used for comparison. The details of the two phonetic dictionaries used for characterization and validation are outlined in the following before discussing the experiments and reporting the results.

A. British English

British English pronunciation rules were inferred from an electronic version of the Oxford Dictionary, called CUVOLAD[2], available from the Oxford Text Archive. The CUVOLAD contains more than 70,000 words, including plural nouns, conjugated verbs, proper names and acronyms. Our training dictionary was obtained by removing proper names and acronyms from the dictionary, reducing to about 60,000 words.

The phoneme set was provided as a list of English IPA symbols with the same ASCII notation used in the CUVOLAD.

The list of English graphemes and the phonogram table were derived from *Orton-Gillingham's* (OG) chart based on the pioneer work of Samuel T. Orton (1925) on dyslexia. The OG chart contains 71 graphemes and the 128 most common grapheme-to-phoneme correspondences. Grapheme ⟨eau⟩, occurring in many words taken from French, and all double consonants were added to the grapheme set. The OG chart was then incremented by adding the new graphemes and their unusual pronunciations found on the CUVOLAD.

The phoneme set, the grapheme set and the phonogram table described so far were provided to the tool and automatically extended to meet completeness requirements. In particular, two new graphemes were added in this phase: ⟨pph⟩ (containing the overlapping graphemes ⟨pp⟩ and ⟨ph⟩) and ⟨ssh⟩ (containing the overlapping graphemes ⟨ss⟩ and ⟨sh⟩).

B. Basic English

Basic English is an artificial subset of English, containing only 850 words, proposed by C. K. Ogden in 1930 as an international second language [12]. A training phonetic dictionary for Basic English was constructed by adding plural nouns and conjugated verbs to the original list of 850 words[3].

[2]CUVOLAD: "Computer Usable Version of the Oxford Advanced Lerner's Dictionary", ftp://ota.ox.ac.uk/pub/ota/public/dicts/710/

[3]http://www.basic-english.org

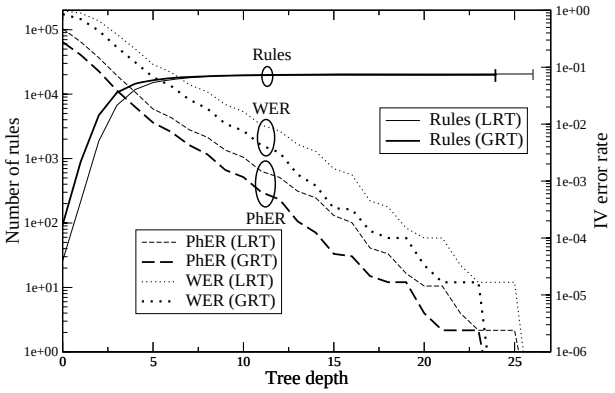

Fig. 3. Efficiency (expressed in number of rules) and accuracy (expressed in number of errors) of LRT and GRT for different depths of the rule trees.

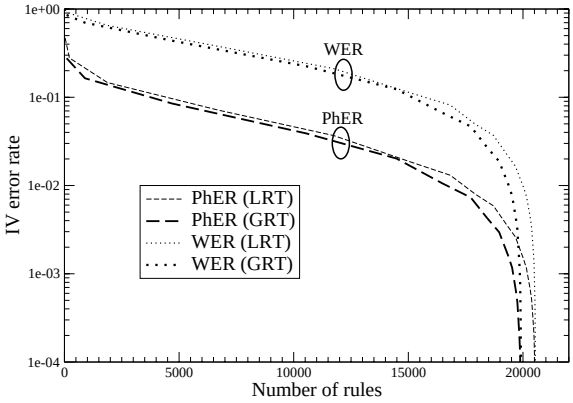

Fig. 4. Accuracy-vs-efficiency tradeoff.

The phonetic transcriptions for the resulting 1490 words were taken from the CUVOLAD.

In this way a reduced training dictionary was obtained defined over the same sets of phonemes, graphemes and phonograms used for the Oxford British English.

C. In-Vocabulary Accuracy

In-vocabulary (IV) accuracy was tested by using the entire CUVOLAD to build the rule trees and by applying them to the same dictionary. Figure 3 shows the number of rules and the IV error rates provided by GRT and LRT for different values of the tree depth. The complete GRT is slightly smaller than the complete LRT, both in terms of rules (19916 against 20548) and in terms of levels (24 against 26). More important, for a given tree depth GRT is more accurate than LRT. Accuracy is expressed in terms of *word error rate* (WER), which is the relative frequency of mispronounced words, and *phoneme error rate* (PhER), which is the relative frequency of wrong phonemes (including substitutions, insertions and deletions).

Limiting the tree depth provides the flexibility of tuning the tradeoff between pronunciation accuracy and rule number, as shown in Figure 4. Although GRT is consistently more efficient than LRT, the main advantage of the grapheme-based approach is the phonetic soundness of the inferred rules. For

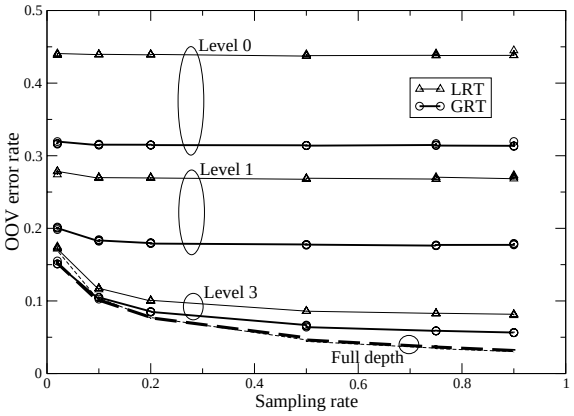

Fig. 5. OOV accuracy as a function of the vocabulary sampling rate for different depths of the rule trees.

instance, the correct pronunciation of 'watch' (i.e., /w//0//tS/) is achieved by LRT by associating ⟨t⟩ with /tS/ and considering ⟨c⟩⟨h⟩ as silent letters. On the contrary, GRT recognizes ⟨tch⟩ as a grapheme to be pronounced /tS/. Similarly, in 'through' (i.e., /T//r//u/) GRT recognizes grapheme ⟨ough⟩ to be pronounced /u/, while LRT arbitrarely associates phoneme /u/ to one of the 4 letters of the grapheme, considering all the others as silent letters.

D. Out-of-Vocabulary Accuracy

Comparative results of OOV accuracy are reported in Figure 5 for different depths of the rule trees: *Level 0* corresponds to a flat tree that takes into account only the grapheme/letter under pronunciation without any context, while *full depth* corresponds to an exhaustive rule tree that takes into account the entire context and makes no pronunciation errors on the training dictionary.

OOV experiments were conducted by using a sample of entries randomly taken from the Oxford dictionary to build the pronunciation rule trees and then applying the rules to the remaining part of the dictionary. Results are plotted as functions of the *sampling rate*, that represents the relative size of the vocabulary used for characterization. The accuracy is expressed in terms of OOV-PhER, that is the rate of the errors made on the pronunciation of each OOV phoneme. Each experiment was repeated three times with different seeds of the random number generator in order to test the statistical significance of the results. The marks representing the outcomes of the three trials of each experiment are almost coincident in Figure 5.

As expected, the higher the sampling rate, the lower the OOV error. In fact, all the curves in Figure 5 have a decreasing trend. Moreover, the effect of the sampling rate (i.e., the slope of the curves) grows with the depth of the rule tree, since larger training sets are required to characterize more complex rules.

The improved accuracy of GRT in comparison with LRT is apparent for the trees having limited depth. This demonstrates that pronunciation rules based on graphemes are more suitable to be generalized than those based on single letters. Full-depth LRT and GRT achieve a comparable OOV accuracy, but GRT requires fewer levels and rules than LRT, as shown in the previous sections.

E. Computational Complexity and Performance

The computational complexity of the tree-construction algorithm depends on the size of the training dictionary, on the target depth of the tree, on the average word lenght and on the average complexity of the pronunciation rules to be inferred. In fact, the tree construction algorithm has an inner loop that iterates on the words of the training dictionary and an outer loop that iterates on the tree levels. The complexity of the body of the inner loop depends, in its turn, on the number of graphemes in the word (since each grapheme may give rise to a new rule), on the depth of the current level (since the previously constructed tree has to be traversed before adding a new rule) and on the regularity of the pronunciation (since new rules need to be added to the tree if and only if they represent exceptions to the rules already encountered). Since the number of new exceptions decreases for increasing tree depths, the lower number of rules to be added at deeper levels compensates for the higher complexity of tree traversal. As a result, the CPU time grows almost linearly with the number of levels and with the number of words. The experiment conducted on a 2.8GHz Pentium 4 with 1GB of RAM running Linux reported a CPU time of 69.8s and 75.7s, respectively, for the construction of complete GRT and LRT from the CUVOLAD. The higher CPU time required for constructing the LRT is mainly explained by the higher number of levels in the complete tree. The average CPU time per level was of about 2.9s, with an average performance of about 48μs per word per level.

It is worth mentioning that tree construction is not a performance-critical task, since it can be performed once and for all and it doesn't impact run-time pronunciation performance.

Using a rule tree to pronounce a word entails a preliminary partitioning of the word in graphemes/letters and a traversal of the rule tree for each of them. Word partitioning is a linear task, whose complexity is proportional to the length of the word, while pronunciation complexity depends on the number of graphemes and on the average depth of the rules needed to pronounce them. The words in the CUVOLAD have on average 8.30 letters and 6.84 graphemes, while the average depth of the pronunciation rules is 1.63 for GRT and 1.89 for LRT. The overall proncunciation complexity per word results in 11.13 for GRT against 15.72 for LRT. In our experiments, the average CPU time required to pronounce a word was 19.3μs for GRT and 20.7μs for LRT. The difference between the two methods is lower in terms of CPU time than in terms of pronunciation complexity because of the flattening effect of parsing and partitioning phases, which take the same amout of CPU time regardless of the rule tree.

F. Application

The proposed methodology was used to compare the empirical pronunciation rules of Oxford and Basic English.

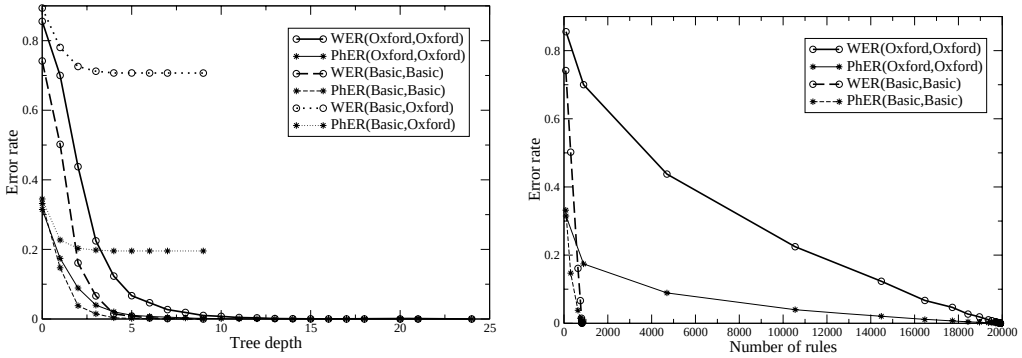

Fig. 6. Experimental results: (a) Pronunciation error rate *vs* depth of the rule tree, (b) Pronunciation error rate *vs* number of rules.

Figure 6.a plots the WER (denoted by circles) and the PhER (denoted by asteriscs) against the maximum depth of the rule tree. Figure 6.b plots WER and PhER against the number of rules in the tree. As expected, the WER is always higher than the corresponding PhER. Each point in the graph represents a different experiment characterized by a given training dictionary, a given context depth and a given validation dictionary.

Solid lines represent the results obtained by using the CUVOLAD both as training dictionary and as validation dictionary. Dashed lines represent the results obtained on Basic English dictionary. The exhaustive rule tree for the CUVOLAD has 24 levels and represents 19,916 rules (to pronounce 410,137 graphemes), while the exhaustive rule tree for the Basic English dictionary has only 9 levels and represents only 822 rules (to pronounce 7,042 graphemes). Hence, Basic English is made of words with a more regular pronunciation than the average Oxford English words.

Dotted lines represent the results obtained by applying to the CUVOLAD the rule trees inferred from the Basic English dictionary. The rule tree that provides no pronunciation errors on Basic English words has a WER of 0.709 and a PhER of 0.196 on the Oxford dictionary.

It is worth noting that different level-0 (i.e., context-independent) pronunciation rules were inferred from the two dictionaries for several graphemes, including ⟨gn⟩ (that is /n/ in Basic English and /g_n/ in Oxford English), ⟨ti⟩ (/S/ and /t_I/), ⟨a⟩ (/eI/ and /&/) and ⟨ew⟩ (/u/ and /j_u/).

IV. Conclusion

This work has introduced a new text-to-speech approach that uses a phonetic dictionary to infer tree-structured pronunciation rules that can be eventually applied to out-of-vocabulary words.

Although the idea of inferring pronunciation rule trees from a phonetic dictionary is not new, the proposed approach improves existing methodologies by using graphemes rather than letters as basic orthographic units. The result is a grapheme-based rule tree that slightly improves the accuracy and efficiency of letter-based methods while significantly improving the phonetic soundness of the inferred rules. The first level of the tree represents the most common context-independent grapheme-to-phoneme correspondences (i.e., phonograms), while subsequent levels represent context-dependent exceptions.

The pronunciation of a word entails the segmentation of the word in graphemes and, for each grapheme, the traversal of the tree from the root up to the node corresponding to the proper context, where the phonemes are annotated. The segmentation is accomplished in linear time during the parsing of the word, by means of an original algorithm that makes use of a complete grapheme set (as defined in the paper) represented as a keyword tree.

Comparative experiments have been performed on Oxford British English to compare the proposed approach with letter-based rule trees in terms of in-vocabulary and out-of-vocabulary performance. Experimental results show the improved quality of grapheme-based rules.

Exhaustive rule-trees provide canonical representations that can be used to compare the pronunciation rules of different languages. This kind of application has been exemplified by comparing the grapheme-based rule trees constructed for Oxford British English and Basic English.

References

[1] A. Aho, Algorithms for finding patterns in strings. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science - Vol. A*. MIT Press / Elsevier, pages 257–300, 1990.

[2] J. Bellegarda, A novel approach to unsupervised grapheme to phoneme conversion. *Pronunciation modeling and lexicon adaptation for Spoken Language (Interspeeech-ICSLP)*, 2002.

[3] N. Chomsky, and M. Halle, *The Sound Pattern of English*, 1968. Harper and Row, New York, USA.

[4] W. Daelemans, and A. van den Bosch, Language-independent data-oriented grapheme-to-phoneme conversion. In J.P.H. van Santen, R.W. Sproat, J. Olive, and J. Hirschberg, editors, *Progress in Speech Synthesis*. Springer, New York, pages 77–89, 1997.

[5] R.I. Damper, and Y. Marchand, A multi-strategy approach to improving pronounciation by analogy. *Computational Linguistics*, 26:195–219, 2000.

[6] R.I. Damper, Y. Marchand, M.J. Adamson, and K. Gustafson, Evaluating the pronunciation component of text-to-speech systems for english: a performance comparison of different approaches. *Computer Speech and Language*, 13:155–176, 1999.

[7] M.J. Dedina and H.C. Nusbaum, Pronounce: A program for pronounciation by analogy. *Computer Speech and Language*, 5:55–64, 1991.

[8] M. Divay and A.J. Vitale, Algorithms for grapheme-phoneme translation for english and french: applications for database searches and speech synthesis. *Computational Linguistics*, 23:495–523, 1997.

[9] T. Dutoit, High-quality text-to-speech synthesis : an overview. *Journal of Electrical & Electronics Engineering*, 17:25–37, 1997.
[10] J. Hochberg, S.M. Mniszewski, T. Calleja, and G.J. Papcun, A default hierarchy for pronouncing english. *IEEE Transactions on Pattern Matching and Machine Intelligence*, 13:957–964, 1991.
[11] J. Lucassen, R. Mercer, An information theoretic approach to the automatic determination of phonemic baseforms *Proc. ICASSP-84 (International Conference on Acoustics, Speech, and Signal Processing)*, 1984.
[12] C.J. Ogden, *Basic English: International Second Language*. Hartcourt, Brace & Jovanovich, New York, USA, 1968.
[13] V. Pagel, K. Lenzo, A.W. Black, Letter to sound rules for accented lexicon compression *Proc. ICSLP-1998 (5th International Conference on Spoken Language Processing)*, 1998.
[14] A. Plucinski, A dynamic context shortening method for a minimum-context grapheme-to-phoneme data-driven transducer generator. *Journal of Quantitative Linguistics*, 13:195–223, 2006.
[15] P.A. Taylor, A. Black, and R. Caley, The architecture of the festival speech synthesis system. *The third ESCA Workshop on Speech Synthesis*, 147–151, 1998.
[16] P. Taylor, Hidden Markov Models for grapheme to phoneme conversion. *The 9th European Conference on Speech Communication and Technology (Interspeeech)*, 2005.
[17] K. Torkkola, An efficient way to learn English grapheme-to-phoneme rules automatically. *Proc. ICASSP-93 (International Conference on Acoustics, Speech, and Signal Processing)*, 1993.