Phonological Parsing for Bi-directional Letter-to-Sound / Sound-to-Letter Generation

by

Helen Mei-Ling Meng

S.M., Massachusetts Institute of Technology (1991)

S.B., Massachusetts Institute of Technology (1989)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

June 1995

Signature of Author . . . .
Department of Electrical Engineering and Computer Science
February 14, 1995

Certified by . . . .
Stephanie Seneff
Thesis Supervisor

Certified by . . . .
Victor W. Zue
Thesis Supervisor

Accepted by . . . .
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

Phonological Parsing for Bi-directional Letter-to-Sound / Sound-to-Letter Generation

by

Helen Mei-Ling Meng

Submitted to the Department of Electrical Engineering
and Computer Science on February 14, 1995 in partial fulfillment
of the requirements for the degree of Doctor of Philosophy

Abstract

This thesis proposes a unified framework for integrating a variety of linguistic knowledge sources for representing speech, in order to facilitate their concurrent utilization in spoken language systems. The feasibility of the proposed methodology is demonstrated on the test bed of bi-directional letter-to-sound / sound-to-letter generation. We present a hierarchical lexical representation which includes information such as morphology, stress, syllabification, phonemics and graphemics. Each of these knowledge sources occupies a distinct stratum in the hierarchy, and the constraints they provide are administered in parallel during generation. A probabilistic parsing paradigm is adopted for generation. The parser is a hybrid of a rule-based formalism and data-driven techniques, and is capable of bi-directional generation. Our training and testing corpora are derived from the high-frequency portion of the Brown Corpus (10,000 words), augmented with markers indicating stress and word morphology. We evaluated our performance based on an unseen test set. The percentage of nonparsable words for letter-to-sound and sound-to-letter generation were 6% and 5% respectively. Of the remaining words our system achieved a word accuracy of 71.8% and a phoneme accuracy of 92.5% for letter-to-sound generation, and a word accuracy of 55.8% and letter accuracy of 89.4% for sound-to-letter generation. The implementation of a robust parsing mechanism shows how generation constraints can be relaxed within the hierarchical framework, in order to broaden coverage and handle nonparsable words. Additionally, a pilot study provides evidence that the framework can be generalized to encompass other linguistic knowledge sources for potential applications in speech synthesis, recognition and understanding.

Thesis Supervisors:
Dr. Stephanie Seneff, Principal Research Scientist
Dr. Victor W. Zue, Senior Research Scientist

Acknowledgments

I wish to express my deepest gratitude to my thesis advisors, Dr. Stephanie Seneff and Dr. Victor Zue, for being outstanding teachers and superb mentors. I thank Stephanie for her enthusiasm and patience in training me as her first doctoral student, and Victor for his altruistic interest in my academic progress. Together they have fostered my professional and personal growth with nurturing guidance, insightful advice, as well as unwavering support and constant encouragement. Working with Stephanie and Victor has been a great pleasure and honor, and I cannot conceive of better thesis advisors. Their profound inspiration will extend far beyond the scope of this work.

I am also grateful to the members of my thesis committee for an expeditious yet careful reading of my thesis. Thanks to Professor Jonathan Allen for sharing with me his experience with the development of the famous MITalk system. I thank Dr. Andrew Golding for taking a keen interest throughout the course of this work, and for his enlightening technical comments and critiques of this thesis. I also thank Dr. Kim Silverman for his stimulating input concerning this research, and for travelling from California to Boston to attend my thesis defense.

I would also like to extend my appreciation to the past and present members of the Spoken Language Systems Group. My thanks go to Dr. Sheri Hunnicutt for many informative discussions about English morphology and her experience with rules generation in MITalk, and for providing the labelled corpus for my experiments; to Dr. Eric Brill for his help with the transformation-based error-driven learning algorithms; to the research staff for many thoughtful comments and feedback about

my work; to Christine Pao and Joe Polifroni for keeping the machines up and running; and to Vicky Palay and Sally Lee for ensuring that everything else runs smoothly. I thank all my fellow students in the Spoken Language Systems Group for their comradeship, and for making school life a lot of fun. Aside from learning from one another and discussing technicalities, we have shared many Friday-afternoon-happy-hours over spectrograms, and together discovered the therapeutic effects of chocolate and tennis. I especially thank my office-mates, TJ Hazen and Raymond Lau, for their moral support and funny jokes, which make the thesis home-stretch much more bearable.

Special thanks also go to all my good friends for many enjoyable times which make student life at MIT (outside the lab) both fun and memorable.

Finally, I express my whole-hearted gratitude to my family — my grandmother, my parents, my brothers and my sister-in-law, for their unconditional love, unfailing support and uplifting encouragement. I thank my parents for providing me with the best education, for instilling in me a strong desire to learn, for their comforting words during times of hardships, and for having faith that I will attain my goals.

This research was supported by ARPA under Contract N00014-89-J-1332, monitored through the Office of Naval Research, and a grant from Apple Computer Inc.

To my family

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Human-machine communication via speech is the shared vision and common goal of many speech researchers. Computers with the power to speak and listen can create a user-friendly, hands-free and eyes-free environment for the user, and the speech medium can provide an efficient and economical mode of transmission. Great strides have been made in many areas of speech research over the past few decades. *Speech synthesizers* [41] have achieved a reasonable degree of clarity and naturalness, and are striving to cover unlimited vocabularies. *Speech recognizers* are now capable of speaker-independent, large-vocabulary, continuous speech recognition. The speech input may either be *read* or *spontaneous*.[1] Vocabulary sizes can range from a few thousand words to tens of thousands of words [63] and efforts to handle out-of-vocabulary words are under way [6], [35]. *Natural language understanding* systems can analyze a recognized sentence to obtain a meaning representation [73]. The semantics are then channelled to the appropriate locations to perform specific actions (such as sav-

[1] Read speech tends to be "cleaner" than spontaneous speech. The latter is characterized by hesitations, filled pauses such as "um" and "ah," false starts (e.g. "I want to fly to Bos-Boston") and ungrammatical constructs.

ing or deleting a file), or to retrieve information (such as airline reservations and city navigation). Interactive information retrieval via speech also requires a *language generation* component for response generation [25]. The combined applications of these four branches of technology, namely, speech synthesis, speech recognition, language understanding and language generation, has brought about the recent advent of *conversational systems* [23] [100] . These systems can carry out a conversational dialogue with the user concerning topics in a restricted domain (or *multiple* restricted domains [28]). The systems accept spontaneous speech as input and respond with synthesized speech as output. They enable the user to solve problems within the designated domain (such as trip planning, weather inquiries, etc.) [24] [28], convey a spoken message with another language via machine translation [94], or learn to read [8] [33] [54].

The development of conversational systems necessitates correct interpretation of spoken input, and accurate generation of spoken output. Decoding the semantics embedded in an acoustic signal, or encoding a message in synthesized speech, involve diverse sources of linguistic knowledge [14] [96]. Amongst these are:

- Signal processing — the transformation of a continuously-varying acoustic speech signal into a discrete form.

- Phonology and acoustic-phonetics — the study of speech sounds, their variabilities as a result of coarticulation, as well as their acoustic characteristics. For example, although the underlying phoneme sequences in “nitrate” and “night rate” are identical, they are realized differently.

- Lexical constraints and word morphology — the knowledge about the composition of words in a language.[2]

- Syntactic information — the rules about grammatical constructs in the forma-

[2] A *morpheme* is the minimal syntactic unit in a language which carries meaning, and a *morph* is the surface realization of a morpheme.

tion of clauses, phrases or sentences from words.

- Semantic information — the meaning of the spoken input. For example, it may be difficult to differentiate between the two sentences "Meter at the end of the street" and "Meet her at the end of the street" based on the acoustics of continuous speech, but they are different both syntactically and semantically.
- Prosodics — the stress and intonation patterns of speech. The location of emphasis in a spoken sentence conveys specific meaning. "I am *flying* to Chicago tomorrow" indicates that *flying* is the means of transportation to Chicago (and not *driving* or others); while "I am flying to *Chicago* tomorrow" proclaims that *Chicago* is the flight destination (and not *Boston* or another city).
- Discourse and pragmatics — the context of the conversation and the rational chain of thoughts invoked. Consider as examples the sentences "It is easy to recognize speech" and "It is easy to wreck a nice beach." Both are semantically reasonable and syntactically well-formed. However, acoustically they are almost indistinguishable. In order to achieve disambiguation, a conversational system needs to resort to information regarding the dialogue context.

These different knowledge sources interact together to modify the speech signal. Word pronunciations, which are a major concern in recognition and synthesis, can be influenced by word morphology, syntax, semantics and discourse. For example, the pronunciation of "unionization" depends on whether the topic of interest concerns "ions" or "unions," which may give the respective derivations "un+ionization" or "union+ization."[3] Semantics is needed for the disambiguation of homonyms such as "see," "sea" and "C". Syntax leads to the different pronunciations between the noun and verb forms of "conduct" (/k ɑ n d ə k t/ and /k ə n d ʌ k t/). Coarticulatory effects in different phonetic contexts and across word boundaries are expressed as phonological rules [62]. Examples include the *flapping* of the /t/ in "water"

[3] This example is borrowed from [21].

(/w ɔ ʕ ɝ/) and the *palatalization* of the /d/ before the word boundary in "did you" (/d ɪ ǰ y u/). Naturally, these rules are found in both synthesizers and recognizers. Prosodics are essentials for conveying [76] and deducing the correct meaning in spoken language; and so is discourse, as can be observed from the examples above. Natural language understanding often involves parsing sentences.[4] The semantic and syntactic information extracted in the process can also be used to trim the recognizer's search space. For example, if a speech recognizer is asked to decode the next word in the sentence "I want to fly from Boston to..." the search algorithm probably should focus on the city names in the vocabulary.

It is therefore obvious that these *interrelated* knowledge sources are indispensable in the development of speech systems, be it synthesis, recognition or understanding. The different types of information, or subsets of them, are often incorporated *independently*, and with ad hoc methodologies, into the components of existing conversational systems. Phonological rules are applied in letter-to-sound generation in speech synthesis [17]. They are also embedded in pronunciation models and networks in speech recognizers [31]. n-gram language models [39] are popular for guiding the search in speech recognizers, because they can be automatically acquired for different tasks with a wide range of perplexities, and are thus more adaptable than finite-state grammars [47] [49]. The recognition outputs may be further re-processed using natural language parsers to provide syntactic analysis and derive meaning. As was shown earlier, semantics and syntax may come into play for reducing the search space of the recognizer, especially for high perplexity[5] tasks (such as those with large vocabularies) where constraints given by the n-gram language models are weak. A lower search complexity should help avoid search errors and maintain high recognition performance. Discourse and prosody have also been used in dialogue management [42].

[4] There also exists systems which attempt to obtain semantics without involving syntactic analysis, see [65] [92].

[5] Perplexity is an information-theoretic measure for the average uncertainty at the word boundary for the next possible words to follow. Later in the thesis we will show how it is computed. A high perplexity signifies a large search space, and a more difficult recognition problem.

We feel that instead of allowing these knowledge sources to reside individually in a conversational system, it is more desirable to model their interrelationships in an integrated framework. The objective of this thesis is to propose a methodology for such integration. The resultant framework should facilitate the concurrent utilization of the knowledge sources, and should be applicable in speech synthesis, recognition and understanding.

1.2 An Integrated Hierarchical Framework for Speech

Having a common framework which integrates all the relevant knowledge sources for speech synthesis, recognition and understanding is advantageous. Not only can it reduce redundancy in development efforts, but also any improvements made in the framework can be inherited by all three tasks. This integration is best exemplified by the human communication system. Our framework is therefore designed to mirror the chain of events underlying the communication between a speaker and a listener, a sequence which has been described as *the speech chain* [20].

When a speaker wants to convey a spoken message to a listener, he first gathers his thoughts, which constitutes the *semantics* of his speech. The semantics is generally coherent with the context of the dialogue, which involves *discourse* and *pragmatics*. The speaker proceeds to configure his message into a linguistic form. He chooses the appropriate words and their *morphological* forms from his vocabulary, and organizes them into sentences and phrases according to the grammar rules or *syntax* of the language. The speech utterance is then formulated in the brain, along with *prosodic* features (pitch, intonation and duration) and *stress* (sentential and word-internal stress) to aid expression. The utterance is then spoken by the coordinated movements of the vocal organs and articulators, producing the *phonetics* of the speech wave which is transmitted from the speaker to the listener. The acoustics of the speech wave is processed by the auditory system of the listener to decode the series of sounds produced by the speaker. From this the listener infers the list of words used, analyzes the structure of the sentence uttered and finally acquires the message conveyed by the speaker. In the event that the sentence contains a new word which is unknown to the listener, learning takes place. The listener secures various properties of the word, such as its meaning, spelling, pronunciation, usage, etc., and subsequently incorporates the word into his vocabulary.

It seems plausible that the design of a unified framework for speech be modeled after the speech chain. We conceive of a grand speech hierarchy with multiple levels of linguistic knowledge sources, grossly ranging from discourse, pragmatics and semantics at the upper levels, through the intermediate levels including prosody and stress, syntax, word morphology, syllabification, distinctive features, to the lower levels of word pronunciations, phonotactics and phonology, graphemics,[6] phonetics and acoustics. The framework should encode not only the constraints propagated along each level of linguistic representation, but also the interactions among the different layers. The hierarchy is illustrated in Figure 1-1. From one perspective, the order of events in speech production is roughly simulated as we descend the hierarchy; while the reverse order as we ascend the hierarchy approximately models the speech perception process. Looking from another perspective, this unified body of linguistic knowledge should be applicable in speech generation/synthesis, recognition and understanding. Furthermore, learning can be achieved if the regularities within the framework can be derived and utilized in generating new structures or representations.

The prime objective of this thesis is to propose such a unified framework of linguistic knowledge sources for multiple speech applications. The test-bed selected for demonstrating the feasibility of our methodology is the task of bi-directional spelling-to-phonemics/phonemics-to-spelling generation.

[6] By “graphemes” we are referring to contiguous letters which correspond to a phoneme.

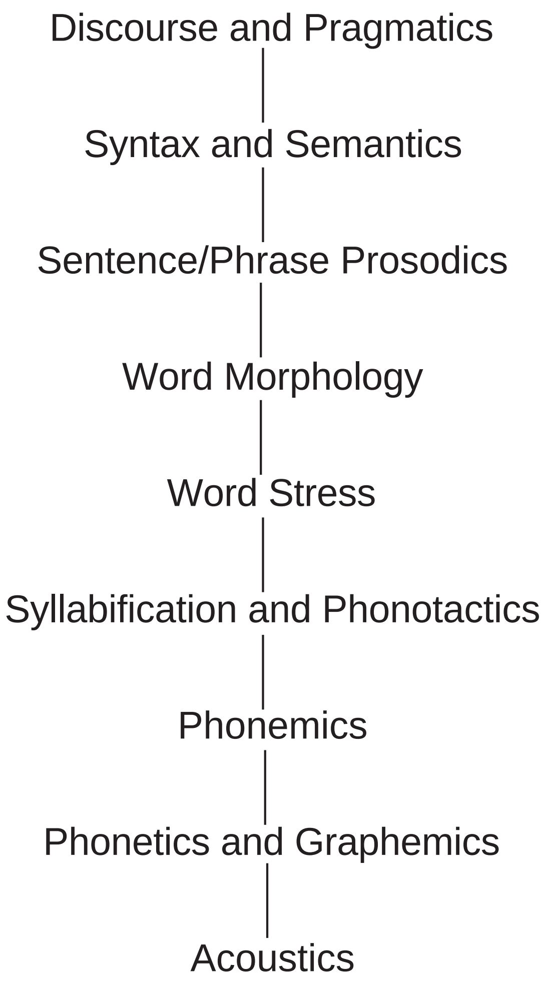


Figure 1-1: A Proposed Grand Hierarchy for Representing Speech

1.3 Spelling-Phonemics Conversion

The scope of this thesis focuses on the task of bi-directional spelling-phonemics conversion. Formalizing the relationship between the spelling and phonemic pronunciation of an English word requires information regarding part of speech, word sense, stress, morphology, phonemics and graphemics. These knowledge sources constitute the substructure in the speech hierarchy (Figure 1-1) which is of immediate relevance to the English word. We have selected as our test-bed the design of representations and algorithms pertaining to the simultaneous/synchronized application of these knowledge sources for bi-directional spelling-phonemics conversion. This task should suffice as apt evidence for the viability of our proposed unified framework, at least on the (smaller) scale of the English word. The thesis will also include preliminary experiments which show the extendability of the implemented framework. The versatility of the framework is implicated by the bi-directionality — the same set of knowledge sources remains pertinent, be it spelling-to-phonemics generation or phonemics-to-spelling generation. In a similar manner, if the grand speech hierarchy in Figure 1-1 is realized, its versatility should transcend to applications in speech synthesis, recognition and understanding.

The bi-directional generation task is also chosen because of its usefulness in handling out-of-vocabulary words in unrestricted text-to-speech synthesis and large vocabulary speech recognition. Text-to-speech synthesizers used as reading machines for the blind, or as interactive voice response for transmission over the telephone, often encounter new words outside their vocabulary. When this happens, letter-to-sound generation becomes the key operating mechanism. Similarly, it is difficult to fully specify the active vocabulary of a conversational system beyond a static initial set. Users should be able to enter new words by providing the spoken, typed or handwritten spellings and/or pronunciations. If only one of the two elements is given, a bi-directional system will be able to automatically generate the other element, and dynamically update the system's vocabulary accordingly.

The development of a bi-directional letter-to-sound/sound-to-letter generator warrants an understanding of the relationship between English orthography and phonology. This will be examined in the following subsection.

1.3.1 Orthographic-phonological Correspondences in English

The English writing system is built from the 26 letters in the alphabet. However, only certain letter sequences are found in English words. Adams [1] [53] noted that,

> *From an alphabet of 26 letters, we could generate over 475,254 unique strings of 4 letters or less, or 12,376,630 of 5 letters or less. Alternatively, we could represent 823,543 unique strings with an alphabet of only 7 letters, or 16,777,216 with an alphabet of only 8. For comparison, the total number of entries in Webster's New Collegiate Dictionary is only 150,000.*

Such a limited set of letters and letter patterns, however, encodes a vast body of knowledge. In fact, the graphemic constraints may very well be a consequence of this. The *alphabetic principle* [69] refers to the occurrence of systematic correspondences between the spoken and written forms of words — the letters and letter patterns found in written English map somewhat consistently to the speech units such as phonemes in spoken English. Chomsky and Halle [11] pointed out that English phonology and morphology are simultaneously represented in the orthography. This suggests that the orthography should exhibit cues which reflect lexical structures like the morpheme. Other lexical structures like the syllable are derived from phonotactic constraints specific to the language, so if written English largely corresponds to spoken English, then syllabic structures should be found in the orthography as well [1] [53].

The way that English orthography corresponds to morphology, syllabification and phonology is fairly systematic, but it also admits many irregularities. Therefore, English has been described as a *quasi-regular* system [53]. To illustrate correspondences in morphology, consider the words "preview" and "decode," which contain the prefix morphs “pre-” and “de-” respectively. Since the meaning of a word is often constituted by the meanings of its morphemes, consistency is preserved in the coining of the words “precompile” and “debug.” However, there are also irregularities such as “pretense” (unrelated to “tense”), and “deliver” (unrelated to ”liver”). To illustrate correspondences in syllabification, there is consistency in the word-pairs “baked”-“faked” and “dies”-“lies,” but inconsistency lies in “baked”-“naked,” and “dies”-“diet.” To illustrate correspondences in phonology, “gave”-“brave” is consistent, while “gave”-“have” is not.[7] Vowels account for many of the inconsistencies in letter-phoneme mappings, since the identity of the vowel in a word is strongly affected by the stress pattern of the word. The stress pattern is in turn dependent on the part of speech of a word, e.g., homographs which can take on two parts-of-speech often have a stress-unstress pattern if they are nouns, and an unstress-stress pattern if they are verbs, as in “record” and “permit.” Another interesting class of exceptional pronunciations arises from high-frequency words [3]. Initial “th” is pronounced as /θ/ (a voiceless fricative) in many words (such as “thin,” “thesis,” “thimble”), but for very frequent words such as the short function words (“the,” “this,” “there,” “those”), “th” is pronounced as /ð/ (a voiced fricative). Similarly, “f” is always pronounced as an /f/ (an unvoiced fricative) except for the single case “of.” Finally, the final “s” in “atlas” and “canvas” is realized as the unvoiced /s/, but for the function words “is,” “was” and “has,” it is realized as the voiced /z/.

As we can see, English orthographic-phonological correspondences seem to operate through the intermediate levels of morphology and syllabification, and contain both regularities and irregularities. Irregularities arise due to the stress pattern of a word, different dialects (e.g. British and American English), lexical borrowings from other languages and spelling reforms, to name a few reasons [53]. Since English is quasi-regular in nature, it seems that a possible way to tackle the spelling-to-pronunciation or pronunciation-to-spelling conversion problems is to capture regularities using rules

[7] These examples are borrowed from [53].

and statistics, while accommodating irregularities using exception dictionaries. Any attempt to determine the orthographic-phonological regularities in English must consider the two important areas of representing and deriving such regularities. In the next section, we will give an overview of the approaches adopted in previous attempts to capture letter-sound regularities for the development of pronunciation and spelling systems.

1.4 Previous Work

1.4.1 Letter-to-Sound Generation

A myriad of approaches have been applied to the problem of letter-to-sound generation. Excellent reviews can be found in [18], [29] and [41]. The various approaches have given rise to a wide range of letter-to-sound generation accuracies. Many of these accuracies are based on different corpora, and some corpora may be more difficult than others. Furthermore, certain systems are evaluated by human subjects, while others have their pronunciation accuracies reported on a per phoneme or per letter basis. Insertion errors or stress errors may be included in some cases, and ignored in others. There are also systems which look up an exceptions dictionary prior to generation, and the performance accuracies of these sytems tend to increase with the use of larger dictionaries. Due to the above reasons, we should be careful when comparing different systems based on the quoted performance values.

The following is a sketch of the various approaches with a few illustrative examples.

1. Rule-based Approaches

   The classic examples of rule-based approaches include MITalk [3], the NRL system [21], and DECtalk [17]. These use a set of hand-engineered, ordered rules for transliteration. Transformation rules may also be applied in multiple passes in order to process linguistic units larger than the phoneme/grapheme, e.g., morphs. The rule-based approaches have by far given the best generation performance. MITalk rules have attained word accuracies ranging from 66% to 76.5% [38] (all phonemes and stress pattern correct). The system Speech Plus Prose 2000 [32] has achieved a performance of 85% word accuracy using only its letter-to-sound rules. Adding exceptions dictionaries helps improve the overall performance noticeably — a 3000-word dictionary with rules gave a 97% word accuracy. In general, rules operate on one-dimensional data structures. There are also rules that operate on two-dimensional data structures, e.g., the Speech Maker formalism [89] developed for Dutch. The two-dimensional rules in the Speech Maker are modelled after the DELTA system [34]. The rules manipulate the contents of a data structure known as the *grid*, which contains streams of linguistic representations synchronized by markers.

Writing rule sets is an arduous process. As the rule set increases in size, the determination of rule ordering and the tracing of rule interactions become more difficult. Furthermore, rules generally have low portability across domains or languages. Therefore, there are also other approaches which try to automatically infer these transcription rules, or the letter/sound correspondences which they represent.

2. Induction Approaches

Induction approaches attempt to infer letter-to-sound rules from a body of training data. The rules follow the form of generative phonology, which gives a letter and its transcription under a specified spelling context. Examples of this approach can be found in [36], [40], [51], [61], [71] and [87]. The following briefly recounts a few of them.

Klatt and Shipman [40] used a 20,000 word phonemic dictionary to create letter-to-sound rules of the form A→[b]/CD _ EF, i.e., the letter "A" goes to the phoneme [b] in the letter environment consisting of 2 letters on each side. If there are rule conflicts, the most popular rule in the conflicting set is used. The computer program organizes the rules into a tree for run-time efficiency, and the system achieved an accuracy of 93% correct by letter.

Lucassen and Mercer [51] designed another letter-pattern learner using an information-theoretic approach. The phonemic pronunciation is viewed as being generated from the spelling via a noisy channel. The channel context consists of 4 letters to the left and right of the current letter, and the 3 phonemes to the left. A decision tree is constructed based on a 50,000 word lexicon, where at each step, the tree includes the context feature with the maximum conditional mutual information.[8] They reported a performance of 94% accuracy per letter on a test set of 5,000 words.

Hochberg et al. [36] devised a *default hierarchy* of rules, ranging from the most general rule set at the bottom to the most specific rule set on top. The bottom-level (Level 1) has 26 general rules, each being a context-independent transcription of a single letter to its most frequent phoneme according to the training corpus. At the next level up (Level 2), each rule includes as context one letter to the left/right of the letter to be transcribed. Level 3 rules are a natural extrapolation — they include up to 2 letters to the left or right of the current letter. Therefore, the rules at level i contain $(i-1)$ letters as context. Training identifies the phoneme $/x/$ in each rule to be the most frequently occurring pronunciation in the training corpus. Each rule has a numerical value computed as its “strength,” which is based on the training corpus statistics. Testing pronounces each letter sequentially, and rule applications are ordered top-down in the hierarchy. Rule “conflicts” are reconciled according to rule strengths and a “majority rules” principle. The system was trained and tested on disjoint sets of 18,000 and 2000 words respectively, and achieved an accuracy of 90% by phoneme. A similar approach was also adopted at Martin Marietta Laboratories [70].

[8] The conditional mutual information between u_1, u_2 and u_3 is defined as $log\frac{P(u_1 \mid u_2,\, u_3)}{P(u_1 \mid u_3)}$.

3. Hidden Markov Models

   Parfitt and Sharman [64] have cast the problem of spelling-to-pronunciation generation in an HMM framework, which has been popular in speech recognition sytems [67] [45]. For the generation task, the HMM has phonemes as its hidden states, with trained transition and observation probabilities, and the orthographic letters as its observed outputs. Based on disjoint training and test sets totalling 50,000 words, the system developed at IBM, UK [64] reported a performance of 85% accuracy per phoneme. Aside from this work, HMMs have also been used for the alignment of orthography and phonemics prior to an inductive learning transliteration procedure for Dutch [86]. Another approach related to HMMs can be found in [52].

4. Connectionist Approach

   A well-known example of this approach is NETtalk developed by Sejnowski and Rosenberg [72]. NETtalk is a neural network that learns the pronunciations of letters. The network consists of three fully connected layers: the input layer takes in a 7-letter context window, where the middle letter is the one to be pronounced and the other six serve as left and right context; the hidden middle layer performs intermediate calculations, and the output layer gives a vector indicative of a phoneme and a stress level (two degrees of stress are included). The network was trained for 5 passes on 1,000 words and tested on a non-disjoint dictionary of 20,012 words. The "best guess"[9] performance was found to be 90% correct by letter. NETtalk was also re-implemented by McCulloch et al. [55] to become NETspeak, in order to examine the effects of different input and output encodings in the architecture, and of the word frequencies on network performance.

   Lucas and Damper [50] developed a system for bi-directional text-phonetics

[9] The dot products between the output vector and the code vector of every phoneme are computed. The phoneme that has the smallest product is the "best guess" output.

translation using two “syntactic” neural networks (SNN) to perform statistical string translation. This system, unlike the others, does not require pre-aligned text-phonetic pairs from training, but instead tries to infer appropriate segmentations and alignments. The first SNN models orthography while the second models phonemics. Training is done in three phases. In the first phase, each SNN allocates a neuron node for the high-frequency substrings in its own domain. In the second phase, transition (bigram) probabilities corresponding to the recurrent connections between neurons within an SNN are estimated. Finally, the third phase learns the translation probabilities between the nodes of one domain and those in the other domain. The activation of a node takes into account all the weighted recurrent connections to that node. The output symbol corresponding to the node with the highest activation is selected as the generated translation. In text-to-phonemics conversion, training and testing on two disjoint 2000-word corpora gave a 66% phoneme accuracy and 26% word accuracy.

5. Psychological Approaches

   Dedina and Nusbaum [19] developed the system PRONOUNCE to demonstrate the computational feasibility of the analogical model. This model is proposed by Glushko [26] in psychology literature, which suggests that humans use a process of analogy to derive the pronunciation for a spelling pattern, as an alternative to the pronunciation-by-rule theory. PRONOUNCE uses a lexical database of approximately 20,000 words. It does not have a training phase. Instead, PRONOUNCE matches each spelling pattern in the test word against *every* lexical entry, and if there are matching substrings, the corresponding phonetic pattern is retrieved to build a pronunciation lattice. After the matching phase, PRONOUNCE traverses the lattice to find the “best path,” using the lengths and frequencies of the subpaths as search heuristics. The system was evaluated on a set of 70 nonsense monosyllabic words, and was found to disagree with human subjects on 9% of the set. Another system modelled after Glushko's theory can be found in [10].

Sullivan and Damper [83] developed a system based on the *dual-route theory* [68], where the duality refers to a set of context-free rules conjoined with lexical analogies. Therefore, Sullivan and Damper's system draws phonemic analogies in addition to orthographic analogies. The orthographic "analogiser"[10] is similar to PRONOUNCE, except that it uses a scoring mechanism based on the text-phonemic mapping statistics, instead of a lexicographic function. The phonemic "analogiser" begins with using a set of context-free rules to generate multiple pronunciations, and these are re-ranked in a way similar to the lexical analogies. The outputs from the orthographic and phonemic analogisers are eventually combined to generate the result.

6. Case-based Reasoning and Hybrid Approaches

   Case-based approaches generate a pronunciation of an input word based on similar exemplars in the training corpus. The TTS system [16] developed at Bell Labs adopts this approach for generating name pronunciations. It operates primarily as a 50K dictionary lookup, but if direct lookup fails, the system will try using rhyming analogies (e.g. "ALIFANO" and "CALIFANO"), perform suffix-exchanges (e.g. "AGNANO" = "AGNELLI" - "ELLI" + "ANO") or append suffixes (e.g. "ABELSON" = "ABEL" + "SON"). If everything fails, then TTS will fall back on a rule-based system named NAMSA for prefix and suffix analysis and stress reassignment.

   MBRtalk [78] [79] is a pronunciation system operating within the memory-based reasoning paradigm. The primary inference mechanism is a best-match recall from memory. A data record is generated for every letter in a training word. Each record contains the current letter, the previous three letters, the

[10] This terminology is adopted from the reference [83].

next three letters, and the phoneme and stress assigned to the current letter. For each letter in the test word, the system retrieves the 10 data records that are most "similar" to the letter under consideration. A special dissimilarity metric is used for the retrieval. Weights are assigned to each of the 10 records according to their dissimilarity to the current letter, whose pronunciation is then determined from the records and their respective weights. Training on 4438 words and testing on 100 novel words gave a performance accuracy of 86% per phoneme. Evaluation by six human subjects gave a word accuracy between 47% and 68%. An extension of this work is found in [80]. Another approach using case-based reasoning can be found in [46].

Golding [29] proposed a hybrid approach based on the interaction of rule-based and case-based reasoning and developed the system ANAPRON. Rules are used to implement broad trends and the cases are for pockets of exceptions. The set of rules is adapted from MITalk and foreign-language textbooks. Each rule records its own set of positive and negative exemplars. In pronunciation generation, the hand-crafted rules are applied to obtain a first approximation to the output, and this is then refined by the case-base if any *compelling* analogies are found. The judgement for *compellingness* is based on the ratio between the positive and negative exemplars in the rules, and the similarity between the test token and the negative exemplars. In this way, rules and the case-base form nice complements. This approach was evaluated on a name pronunciation task, with a case-library of 5000 names, and a separate set of 400 names for testing. The percentage of acceptable pronunciations was measured and compared with NETtalk and other commercial systems (from Bellcore [77], Bell Labs [16], and DEC [17]). ANAPRON performed significantly better than NETtalk in this task, yielding a word accuracy of 86%, which is very close to the performance of the commercial systems.

Van den Bosch et al. [88] experimented with two data-oriented methods for grapheme-to-phoneme conversion in Dutch. The first variant, known as *instance-based learning* (IBL), is a form of case-based reasoning. During training it constructs records of letters surrounded by different graphemic windows, the corresponding phonemic transcriptions and statistics. Testing involves retrieving the record most similar to the test letter, using an information-theoretic metric, and taking the most frequent phonemic transcription of the record as the generation output. The second variant is a table-lookup method. For each letter in the training data, the table stores the minimum context required to arrive at an unambiguous transcription, up to five letters to the left and right of the current letter (a 5-1-5 grapheme window). Testing is essentially a table retrieval process. However, if retrieval fails to find a match, the test procedure is supported by two "default tables," which use grapheme windows of 1-1-1 and 0-1-0 respectively. The reference also suggested the use of IBL to replace the default tables. This idea is similar to Golding's method in that it is also a hybrid — between the table and the case-base, instead of rules and the case-base. Using the table method on English transliterated (18,500 training words and 1,500 testing words) gave a 90.1% accuracy per letter.

1.4.2 Sound-to-Letter Generation

The development of spelling systems is a task rarely undertaken. We know of three approaches that have previously been adopted:

1. A Combined Rule-based and Inductive Approach

   The rule formalism in generative phonology is also used in generating spelling rules [95]. Two lexicons of respective sizes 96,939 and 11,638 were transcribed with one-to-one phoneme-to-grapheme matches, using the /null/ phoneme and "null" letter when necessary. Upon analysis of the lexicons, it was felt that there was insufficient consistency for a rule-based system. Therefore, each lexicon was split according to word phonemic length, and their respective rule sets were found as a function of phoneme position, in addition to the local phonemic context. Therefore, the format of a typical rule is:

$$\mathit{Rule}:\ \mathit{num},\ \mathit{pos},\ P_0,\ \mathit{phoneme\ context},\ G$$

where num is the number of phonemes in the pronunciation, pos is the position of the current phoneme P_0, which maps to grapheme G under the specified phonemic context (up to two phonemes on either side P_0). For example, the rule {5, 3, /ɑʸ/, P_{-1} =/b/ and P_{-2} =/ɑ/, "I"} states that the phoneme /ɑʸ/, when preceded by the di-phoneme /ɑ b/, generates the grapheme "I" (e.g. in the word "abides", pronounced as /ɑ b ɑʸ d z/).

The rules are searched sequentially, given the word length and phonemic position, in the general order of increasing phonemic context: (i) no neighboring phonemes, (ii) one phoneme on the right, (iii) one phoneme on the left, (iv) one phoneme on each side, (v) two phonemes on the right and (vi) two phonemes on the left. The search proceeds until a unique grapheme is found. If there are none, the system is considered to encounter a failure. Each rule set is tested on the lexicon used for its generation. Word accuracies on the small and large lexicons are 72.4% and 33.7% respectively. Another set of experiments were conducted whereby the system can revert to a set of "default" rules upon failure. These rules are manually written with reference to the lexicons. Accuracies rose to 84.5% and 62.8% for the small and large lexicons respectively.

2. Hidden Markov Models

   HMMs have also been used by Alleva and Lee [4] for acoustics-to-spelling generation. The problem is formulated roughly as an inverse of the previous application of HMMs on spelling-to-pronunciation generation — the surface form is the acoustic signal, and the underlying form is the orthography. Therefore the HMMs model the relationship between the acoustics and orthography of 15,000 continuously spoken sentences. Phonetic transcription is totally bypassed, which makes the problem more difficult. Quad-letter models are used to represent the letter under consideration, two left letters and one right letter. These are used in conjunction with a five-gram letter language model in the Sphinx recognition system [45]. Testing on a disjoint corpus of 30 embedded and end-point detected words gave a 72.7% letter accuracy, 39.3% letter error rate and 21.2% string accuracy. Since the letter accuracy and error rate add up to more than 100%, it is assured that insertion errors were omitted for letter accuracy.

3. Connectionism

   The aforementioned Syntactic Neural Network system [50], which is the only reversible system we have found in the literature, gave a 71% letter accuracy and 23% word accuracy when trained and tested on two disjoint 2000-word corpora.

1.4.3 Summary of Previous Approaches

Tables 1.4.3 and 1.4.3 summarize the two previous subsections.

1.5 Thesis Goals

In essence, the common thread running behind most automatic generation systems is the acquisition of transcription rules or swatches of letter/phoneme patterns, which enfold local context for letter/sound generation. These entities (rules or patterns) can either be written by linguistic experts or inferred from training data. If the window of context involved is narrow, the entity tends to have high data coverage, i.e., it is applicable to many test words. However, entities with narrow context windows also have a lot of ambiguities. Disambiguation needs long-distance constraints, which leads to the widening of the context windows. The corresponding rules/patterns hence

Approach	Example Systems	Corpora	Word Accuracy	Phoneme Accuracy
Rule-based	MITalk	200 (test)	66%-77%	—
	SPP (rules only)		85%	—
	SPP (rules and exceptions)		97%	—
Induction	Klatt & Shipman	20K	—	93% per letter
	Lucassen & Mercer	50K (train) 5K (test)	—	94% per letter
	Hochberg et al.	18K (train) 2K (test)	—	90% per phoneme
HMM	Parfitt & Sharman	50K (train and test)	—	85% per phoneme
Connectionist	NETtalk	20K (train) 1K (non-disjoint test)	—	90% per letter
	Lucas & Damper (SNN)	2K (train) 2K (test)	38%	71% per phoneme
Psychological	PRONOUNCE	70 nonsense syllables	91%	—
Case-based Reasoning	MBRtalk	4K (train) 100 (test)	47-68%	86%
Case and Rule Hybrid	Golding (ANAPRON)	5K (train) 400 (test)	86%	—

Table 1.1: Previous Approaches for Letter-to-sound Generation

Approach	Example Systems	Corpora	Word Accuracy	Letter Accuracy

Rule-based and Inductive Hybrid	Yannakoudakis & Hutton	12K(train and test)
		97K(train and test)
			72%
			34%
				85%
				63%

HMM	Alleva & Lee	15K sentences (train) 30 embedded words (test)	21%	61%
Connectionist	Lucas & Damper (SNN)	2K (train) 2K (test)	23%	71%

Table 1.2: Previous Approaches for Sound-to-letter Generation

become more specific. Specificity implies low data coverage, and the large number of distinct, specific rules or cases often poses computational problems. Decent generation performance and coverage demands a good mix of general and specific rules and cases. The use of this mixture entails elaborate efforts in reducing redundancy and resolving conflicts, especially when the size of the rule-set or case library is large.

Phoneme accuracies of the data-driven systems generally hover around the low 90 percentages.[11] This roughly translates to $(0.9)^6 = 53\%$ word accuracy, if we assume that an average word is 6 letters long, and the probability of pronouncing each letter correctly in a word is independent of the other letters. It is therefore obvious that there is quite a wide performance gap between the automatic systems and systems using hand-crafted rules, which typically can attain word accuracies in the 80-90% range. This tacitly reflects the insufficiency of local context for generation. It is mainly the rule-based approaches which apply suprasegmental constraints to some significant extent. Suprasegmental rules operate on larger linguistic units, e.g. morphs and syllables, to enforce long-distance constraints concerning morphology[12] and syllable stress patterns in a word. These rules also tend to be executed in a sequential manner, adding further complexity to the existing rule specification.

Reiterating our earlier statement, this thesis adopts a novel approach in spelling-phonemics conversions which differs from the ordered transformations and local pattern matchers. Relevant knowledge sources, including those beyond the local letter/phoneme context, are united in a hierarchical framework, where each knowledge source occupies a distinct stratum. All the constraints beneficial to the generation task at hand (from long-distance constraints for suprasegments to short-distance constraints for transcription) are administered *in parallel*.[13] The advantages of this for-

[11] The accuracies quoted amongst the different systems should not be strictly compared, because some are measured on a *per letter* basis; others on a *per phoneme* basis, and with different data sets. We will address this later in the thesis.

[12] Morphotactics refers to the positional constraints for the morphs in a word. In general, the location of morph boundaries are considered to be very important in letter-to-sound generation, because generation rules which operate within a morpheme often break down across morph boundaries.

[13] This idea shares similarities with the synchronized rules in the Speech Maker formalism [89] for

malism are three-fold:

1. The higher strata in the hierarchy embody longer-distance constraints. These provide additional information to the limited context used in local string matches, and may also help eliminate the large number of "specific" transcription rules.

2. Interactions between the variable-sized units from different knowledge sources (morphs, syllables, phonemes, graphemes, etc.) are harnessed in the hierarchy framework. Hence, one can avoid the tedium of tracking rule interactions and resolving rule conflicts in the determination of a rule order. The framework also offers a thorough description of the English word at various degrees of resolution.

3. Serial, transformational rules generate arbitrarily many intermediate representations between the input form and the output form. Once a rewrite-rule is applied, the identity of the representation prior to the rewrite is lost. Therefore, transformation from the input form to the output form is irreversible. Contrarily, the integrated framework is inherently bi-directional. The hierarchical framework preserves the same constraints exercised in *both* letter-to-sound and sound-to-letter generation. Consequently, the new formalism should be more efficient and economical.

Generation is performed in a parsing framework, which is suitable for providing a hierarchical analysis of the input. The parser design is a hybrid which combines the merits of a knowledge-based approach (i.e. high performance accuracy) with those of a data-driven approach (i.e. automation and robustness), by incorporating simple and straightforward linguistic rules into a probabilistic parser. The *probabilistic* parsing paradigm is preferred for four reasons: First, the probabilities serve to augment the

text-to-speech synthesis, and the two-level rules found in the PC-KIMMO system for morphological analysis [5].

known structural regularities that can be encoded in simple rules with other structural regularities which may be automatically discovered from a large body of training data. Secondly, since the more probable parse theories[14] are distinguished from the less probable ones, search efforts can selectively concentrate on the high probability theories, which is an effective mechanism for perplexity reduction. Thirdly, probabilities are less rigid than rules, and adopting a probabilistic framework allows us to easily generate *multiple* parse theories. Fourthly, the flexibility of a probabilistic parser also permits us to automatically relax constraints to attain better coverage of the data.

In short, the goals of this thesis are :

- to demonstrate the feasibility of assembling and integrating multiple linguistic knowledge sources (lying within the scope of the English word) in a hierarchical framework, and
- to illustrate the versatility and parsimony of this unified framework in terms of the *bi-directionality* in spelling-phonemics conversion via a probabilistic parsing paradigm.

1.6 Thesis Outline

In this introductory chapter, we have given a brief overview of spoken language research, placing particular emphasis on the interdisciplinary aspect of the problems involved. We feel that it is desirable to combine and coordinate the suite of knowledge sources to form a coherent framework for the various speech components, and will proceed in the following to describe our attempts to achieve this goal. The rest of the thesis is organized as follows:

Chapter 2 describes the lexical representation which we have created for the English word. It is a hierarchical representation designed to integrate different levels of

[14] A parse theory suggests a possible way of parsing a word.

lingustic representation, namely, morphology, stress, syllabification, distinctive features, phonemics and graphemics. Therefore, a collection of variable length units such as morphs and syllables are used, in addition to phonemes and letters.

Chapter 3 explains the bi-directional, synthesis-by-analysis algorithm used to accomplish our generation tasks. It is based on a probabilistic parsing paradigm, entitled the Layered Bigrams, which is used in accordance with the hierarchical lexical representation. The parser is a hybrid of rule-based and data-driven strategies. Details about the training phase, the testing phase, as well as the search mechanism, will be provided.

Chapter 4 presents information about the data used for our experiments, and the evaluation criteria by which we measure our performance. Results will also be reported for both letter-to-sound and sound-to-letter generation, followed by an analysis of some generation errors.

Chapter 5 lists a series of experiments which illustrate the advantages of using the hierarchical framework by comparing it with an alternative "non-linguistic" analysis based on variable length letter/phoneme n-grams. The hierarchical representation supplies a collection of constraints which together enhance efficiency and accuracy in generation. In addition, it is a compact representation, requiring few system parameters as it promotes a high degree of sharing among different words.

Chapter 6 addresses a major issue of concern, parser coverage, because a non-parsable word spelling or pronunciation does not yield any generated output. We have implemented a "robust" parser, which is capable of relaxing certain constraints to handle the problematic words and broaden coverage.

Chapter 7 examines the extendability of the hierarchical layered bigrams framework. It is a small step towards an existence proof that this framework can encompass other linguistic levels in the full-fledged speech hierarchy conceived in Figure 1-1. We have added a phone layer to the layered bigrams framework, and shown how it is possible to automatically capture phonological rules with probabilities trained from hand-labelled data.

Finally, Chapter 8 summarizes the thesis and discusses future extensions of the proposed framework, as well as possible applications.

Chapter 2

The Lexical Representation

This chapter presents the lexical representation which we have designed for the bidirectional generation tasks. The knowledge sources which have immediate relevance to graphemic-phonemic mappings in the English word, and which form a subhierarchy in Figure 1-1, are united and integrated into a succinct description. This description also forms the infrastructure for generation, which is explained in the next chapter.

2.1 Integration of Various Linguistic Knowledge Sources

The lexical representation is a hierarchical structure which assimilates the relevant linguistic knowledge sources to capture the orthographic-phonological correspondences in English. Each level of linguistic representation is composed of a small set of lexical units, each serving a unique descriptive role. The several distinct and well-defined layers in the lexical representation preserve the ordering of the knowledge sources in Figure 1-1. The layers are defined from top-to-bottom in Table 2.1.[1]

[1] Phonemes are enclosed in / /, graphemes in #[ ], and phones in [ ] — as will be seen in Chapter 7. The categories for each layer are shown in Appendices A through F. If we define a column history

Layer	No. of Categories	Examples
1. Top Level	1	WORD
2. Morphs	5	PREF, ROOT, ROOT2, SUF
3. Stress	8	SSYL1, SSYL2, SYL
4. Subsyllabic Units	5	ONSET, NUCLEUS, CODA
5. Broad Classes	7	STOP, NASAL
6. Phonemes	52	/æ/, /k/
7. Graphemes	205	#[ck], #[gue], #[ai], #[kn]

Table 2.1: Table showing the different layers in the lexical representation, the number of categories in each layer and some example categories.

The size of lexical units decreases as we descend the hierarchy. A word consists of one or more morphs, and different words may share the same morph(s). The same relationship is found between morphs and syllables. A given syllable is often identified with its level of stress, and each syllable has one or more syllable parts (or subsyllabic units). The syllable structure provides tactics for phonology, which is why the manner, place and voicing features are placed beneath the subsyllabic unit level. Letters or graphemes are located at the bottom because they often have direct correspondences with phonemes. The phonetic layer is considered to occupy the same level as the graphemic layer in this hierarchical ordering.

The top level currently consists of a generic [WORD] category, but it can conceivably be used to encode semantic information such as word sense, or syntactic information such as part-of-speech or tense.[2] Semantic and syntactic characterization may change the pronunciations of words. For example, "bass" may be pronounced as /b e s/ or /b æ s/, depending on whether we are referring to music or a fish. Homographs like "permit" and "record" are pronounced with complementary stress patterns, depending on the part of speech (noun or verb forms). Similarly, "read"

to be a feature vector with seven categories, one from each level shown in the table, then there are fewer than 1,500 unique column histories in our training data.

[2] Other information may also be included, such as the origin of loan words, should we decide to model words of different origins as separate categories.

may be pronounced as /r i d/ or /r ɛ d/, depending on the tense.

The second layer, morphology, embodies some morphophonemic effects. Letter-to-sound mappings which are consistent within morphs may be altered across morph boundaries. For example, the letter sequence "sch" in "discharge" is pronounced differently from that in "scheme." The former "sch" sequence overarches a morph boundary between "s" and "c" which separates the prefix morph "dis-" and the root morph "-charge," while the latter sequence belongs to a single root morph. Another similar example is provided by the word "penthouse," where the letter sequence "th" is not realized as a medial fricative, due to the presence of a morph boundary in between the two letters. Morph composition also brings about spelling changes [59].[3] For instance, the final "e" in the suffix "ize" of the word "baptized" is redundant with the "e" of the inflectional suffix "ed," and so one of the redundant letters is dropped. Other cases of deletion are evidenced in the word "handful," derived from "hand" and "full," and in the word "handicap," coming from the three words "hand," "in" and "cap." There are also examples of insertions due to morph combinations, such as the gemination of "g" in "begged," which did not appear in the root "beg."

The third layer is a sequence of stressed and unstressed syllables. Stress strongly affects the identity of the vowel in a syllable, as can be seen by comparing the words "finite" and "infinite." The first syllable in "finite" is stressed and contains the diphthong /ɑʸ/, but the corresponding syllable in "infinite" becomes unstressed, and the diphthong reduces to a front schwa /ɪ/. In addition, stress affects the placement of syllable boundaries, which is illustrated by the words "fabric" and "fabrication." The letter "c" in "fabric" forms the *coda* of the second syllable. However, upon the addition of a "stress-affecting suffix" such as "-ation,"[4] "c" has been moved to become the *onset* of the following stressed suffix syllable. This movement occurs

[3] These spelling change rules, however, have not been explicitly incorporated in our lexical representation.

[4] When a morph is extended by a "stress-affecting" suffix, the syllable preceding the suffix is forced to become unstressed.

if we abide by syllabification rules such as the Maximal Onset Principle and Stress Resyllabification.[5] Furthermore, stress information is necessary for the application of certain phonological rules. Schwa elision requires a stressed-unstressed-unstressed pattern. For instance, if we consider words such as "difference" and "opera" to have three syllables with the stressed-unstressed-unstressed stress pattern, the application of the schwa deletion rule will reduce the words to two syllables, dropping the /ɪ/ in the middle syllable. Similarly, the flapping rule requires a falling stress pattern, and therefore the second /t/ is flapped in "strategy" but not in "strategic."

The next couple of layers, with subsyllabic units in the fourth and broad manner classes in the fifth, jointly define the syllable structure of the word. The morph layer is deliberately positioned above the syllable layer. This is because syllable theory implicitly assumes that a given syllable can transition to any other syllable. However, since there are only a finite number of prefixes and suffixes, morphology provides constraints for the syllables. In addition, precise syllable boundaries are often hard to locate. For example, the syllable boundary in "monkey" may be placed between the phonemes /ŋ/ and /k/, or between /k/ and /i/. In these circumstances, we may be able to utilize morph boundaries to aid placement of the syllable boundaries. According to our data,[6] the word "monkey" is composed of the root morph "monk-" and the suffix "-ey." Consequently, the selected syllabification for the word "monkey" places the syllable boundary between the phonemes /k/ and /i/. This is shown in Figure 2-1.

The fourth layer, syllable parts, also provides tactics for the two successive layers of distinctive features [22] [81]. The sequence of broad classes (manner features) in the fifth layer bears the Sonority Sequencing Constraint. This rule states that the relative prominence or "vowel-likeness" of a sound decreases as we move from the

[5] The Maximal Onset Principle states that the number of consonants in the onset position should be maximized when phonotactic and morphological constraints permit, and Stress Resyllabification refers to maximizing the number of consonants in the stressed syllables.

[6] The morphological decomposition of our data is provided by Sheri Hunnicutt [59].

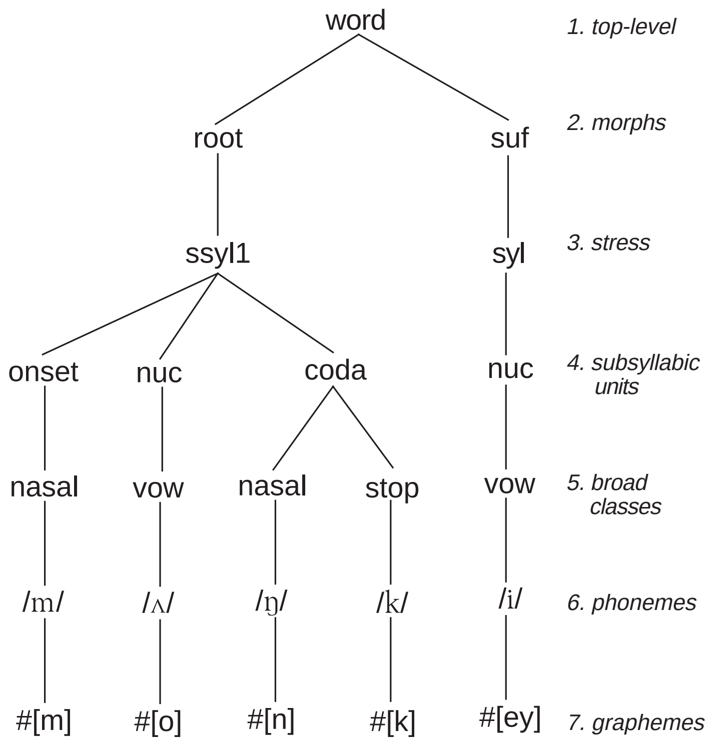


Figure 2-1: Lexical representation for the word "monkey" — shown here in a parse tree format.

syllable nucleus towards the syllable margins. Place and voicing features are encoded as phonemes in the sixth layer.[7] We think that it is important to include a layer of phonemes for three reasons - (i) it serves to represent the pronunciation of a word, (ii) the sequential ordering of the phonemes allow us to infer phonotactic constraints as well as the ongoing articulatory processes during pronunciation, and (iii) it preserves the language-dependent characteristics since the inventory of phonemes varies from one language to another. In this work we have an inventory of 52 “phonemes”, as listed in Appendix E. The set includes both stressed and unstressed counterparts of the vowels /i/, /o/ and /u/, as well as some pseudo-diphthongs such as /ɔ r/ and /y u/. The advantage of having these pseudo-diphthongs will become obvious in the next chapter when we present our probabilistic parser. Essentially, the parser makes use of bigram constraints, but the use of diphthongs and pseudodiphthongs indirectly incorporates “trigram” constraints into the parser. Furthermore, if our framework is to be applied to segment-based speech recognition systems, the use of pseudo-diphthongs may pose an additional advantage because segment-based speech recognizers often have difficulty delineating the boundary in between a phonemic or phonetic pair such as /ɔ r/, /y u/, /ɑ l/ and /o l/, where only gradual acoustic transitions are found.

Finally the seventh layer of terminal nodes represents the letters in a word spelling. The graphemes acquired from the training data are listed in Appendix F. We will show in Chapter 7 that if the terminal layer is used as a *dual* representation of letters and phones, this hierarchical lexical representation can be used to capture phonological rules between the preterminal and terminal layers of phonemes and phones.

[7] In some recently published experiments on tree-based unit selection for English speech synthesis, Sagisaka et al. [91] confirmed that syllabic stress, as well as the place and manner of voicing of the preceding phonemes, are important context variables.

2.2 Some Examples

In this section, we provide the lexical representations of some words in our corpus for illustrative purposes.

Figure 2-2 shows the description for the word "dedicated." The sequence of morphs — prefix, root and suffix, constitutes the morphological composition. The suffix contains an inflectional suffix syllable.[8] Primary stress is placed on the first syllable, secondary stress on the third, while the second and the fourth syllables are reduced. The special "moved-onset" M-ONSET category is found in the fourth layer. It signifies that the letter "c" should belong to the root "dic,"[9] but has become a moved onset of the next syllable due to the Maximal Onset Principle and the Stress Resyllabification Principle. In the terminal layer, we see the special terminal category #[*], which is a graphemic "place-holder" introduced to maintain consistency between the representations of the words "dedicated" and "dedicate" (compare Figures 2-2 and 2-3). Consistency enhances structural sharing among the representations of different words. Furthermore, the effects of morph composition on spelling changes, such as the deletion of one of the redundant "e" letters from the suffix "-ate" and the inflectional suffix "-ed," is expressed indirectly by #[*].

Figure 2-4 shows the representation for the word "taxes." This is an instance where a letter can map to more than one phoneme (i.e., "x" maps to /k/ and /s/). Again, the graphemic place-holder #[*] is used to handle such examples.

In general, roots of words are defined as having one or two syllables. The second syllable of the root is categorized as [ROOT2]. It often consists of a single vowel phonemically, optionally followed by a liquid, and is frequently unstressed. Figure 2-5 shows an exemplar occurrence of the [ROOT2] category in "hero," and Figure

[8] Inflectional suffixes are generally used to mark tense and number (i.e. plural forms). Examples include "-ed" and "-s." They are placed at the syllable level because they may take the form of an unstressed syllable or a subsyllabic unit.

[9] According to Webster's New World Dictionary, the root of "dedicated" is "-dic-," which is derived from the Latin word "dicare."

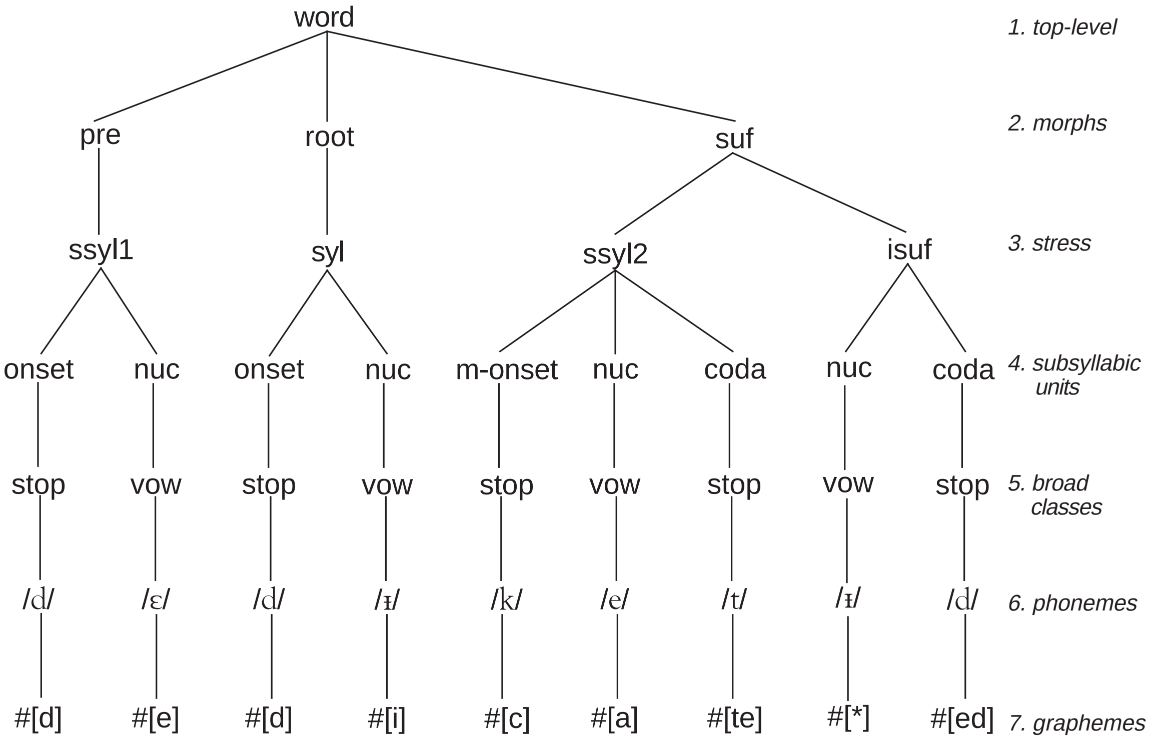


Figure 2-2: Lexical representation for the word "dedicated" - shown here in a parse tree format, and with the different linguistic layers indicated numerically.

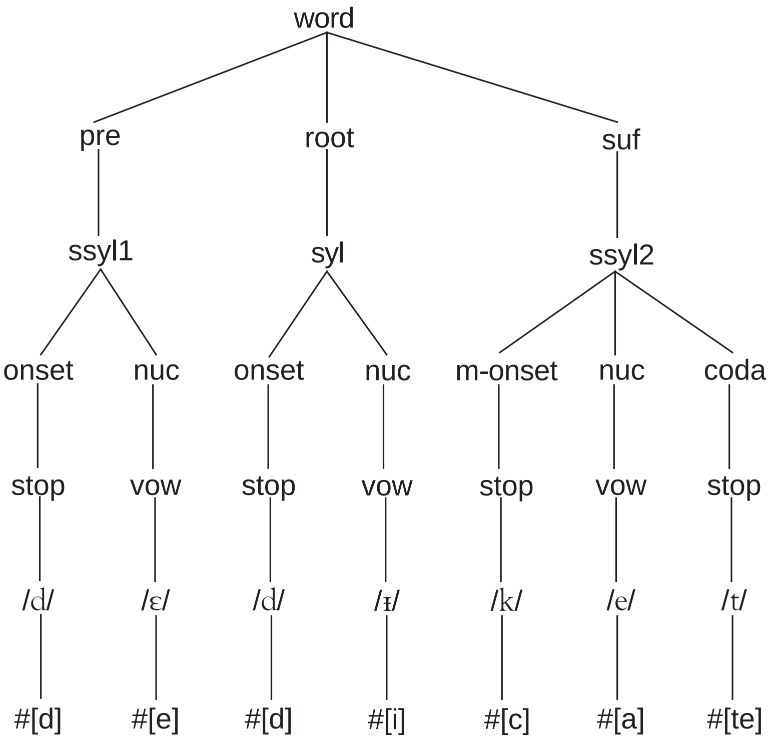


Figure 2-3: Lexical representation for the word “dedicate” - shown here in a parse tree format.

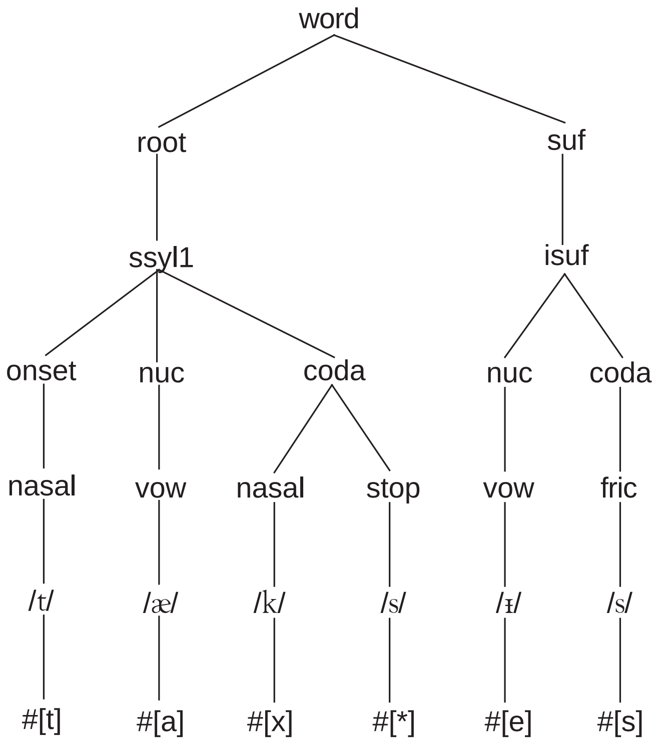


Figure 2-4: Lexical representation for the word “taxes” - shown here in a parse tree format.

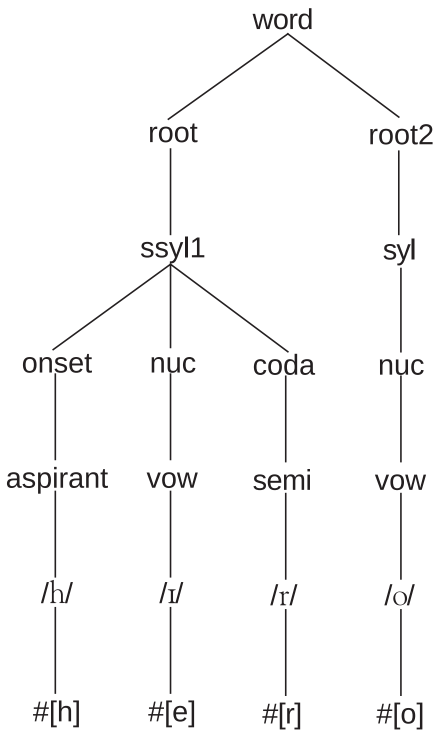


Figure 2-5: Lexical representation for the word "hero."

2-6 shows the onset movement for /r/, incurred in "heroic" when the stress-affecting suffix "-ic"[10] is appended.

Prefixes may sometimes be formed by adjoining a "connecting vowel" to a root or a whole word structure, as exemplified by "accelerometer." In the lexical representation for this word (Figure 2-7), the derivation of the prefix "accelero-" is preserved, with the intent of promoting structural sharing among words. The special categories, [JOIN], [JOIN-SSYL] and [JOIN-VOW] are created for the description of the connecting vowel "-o-", which is stressed, and pronounced as /ɑ/.

The representations of compound words are simply the merger of the individual word representations. The parse tree for "headlight" is depicted in Figure 2-8. The first root, "head," acquires primary stress, and the second root, "light," carries

[10] The attachment of the suffix "-ic" causes stress to be placed on the preceding syllable.

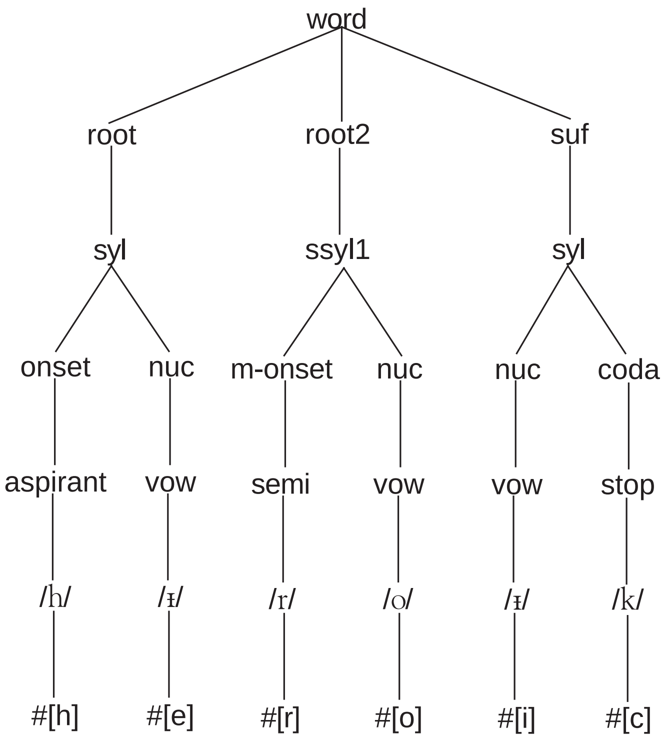


Figure 2-6: Lexical representation for the word “heroic.”

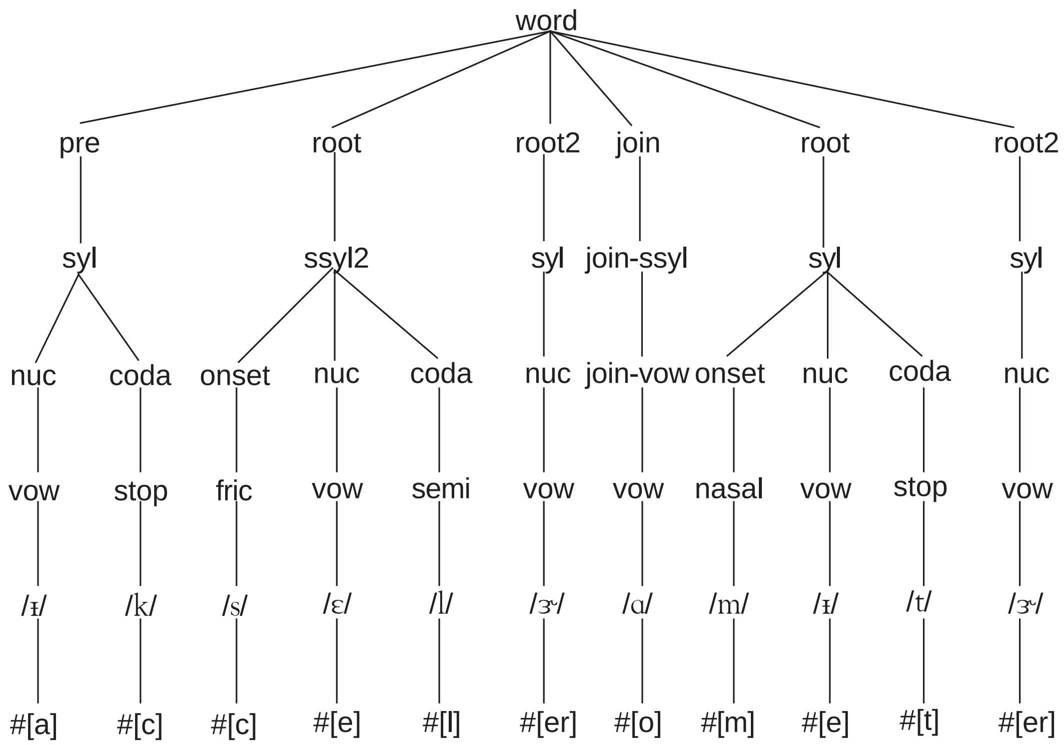


Figure 2-7: Lexical representation for the word "accelerometer."

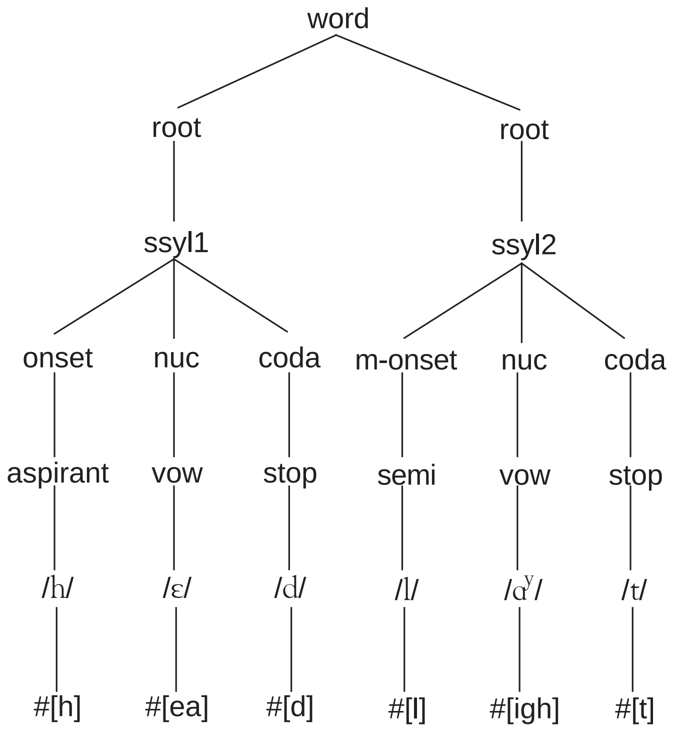


Figure 2-8: Lexical representation for the word "headlight."

secondary stress.

The proper names which are present in our experimental corpus pose a problem. The elements involved in the morph composition of proper names is quite dissimilar to those of common English words. Our representations for proper names adopt morphological decompositions which are concocted with preferences towards consistency and structural sharing. Figures 2-9 and 2-10 portray two examples. The latter example illustrates that a root with multiple syllables is treated as multiple roots.

A number of conventions are held while creating the hierarchical parse trees for particular words. Amongst these are the words ending with the suffix "-ism." The letters show a vowel for "i" being followed by a fricative and then a nasal, e.g. "buddhism," "capitalism," "optimism," etc. If these three letters constitute a single syllable, the Sonority Sequencing Principle will be violated. In order to avoid this problem,

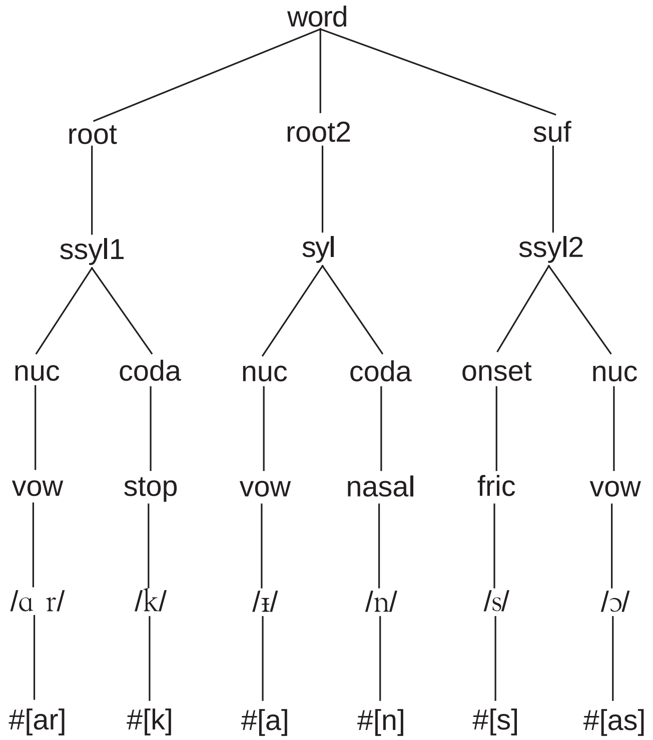


Figure 2-9: Lexical representation for the name “Arkansas.”

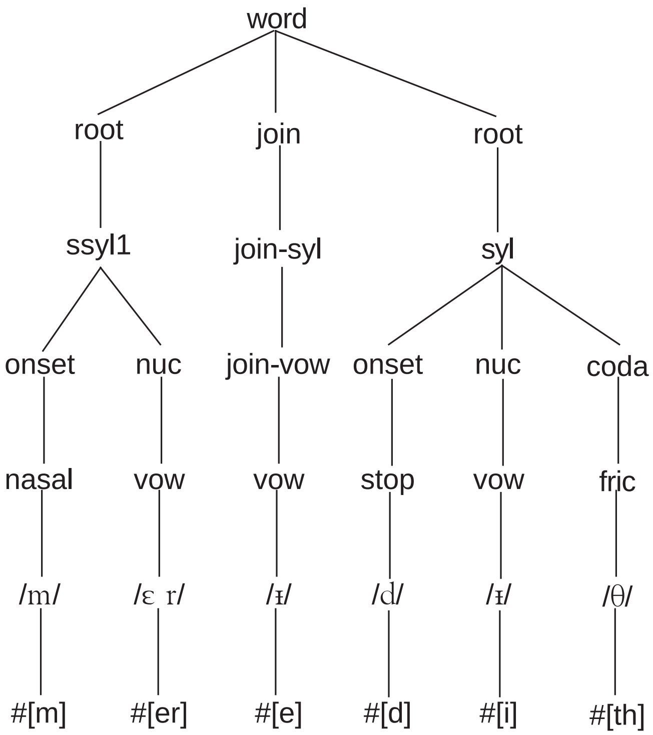


Figure 2-10: Lexical representation for the name “Meredith.”

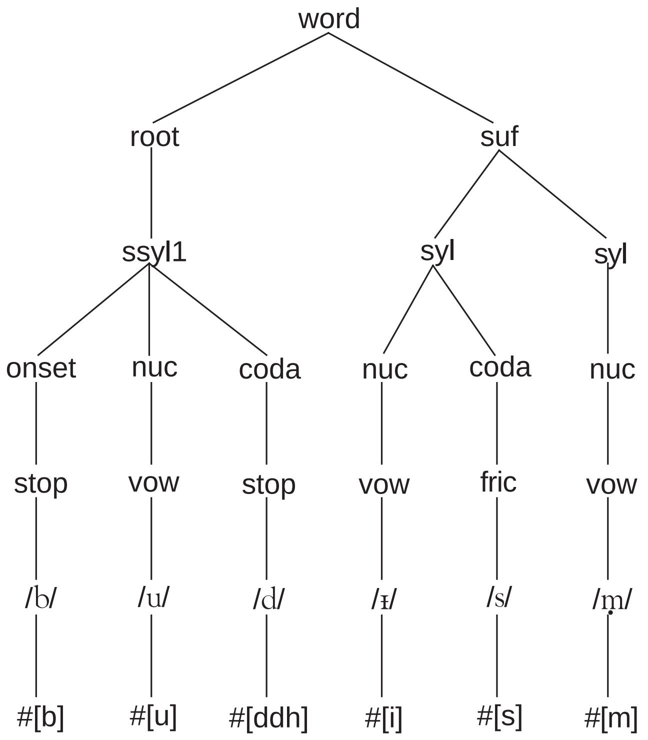


Figure 2-11: Lexical representation for the word "buddhism."

the letter "m" is isolated to become a separate syllable formed by the "syllabic m" phoneme (/m̩/), as shown in Figure 2-11.

Another principle concerns the use of a *pure* phonemic form in the parse tree. For example, the parse tree for "national" (Figure 2-12) maps /s/ to "t" and /y/ to 'i',[11] instead of /š/ to "ti." The selected phonemes are closer to the underlying phonemic correspondence for the letters, i.e. a strident fricative for "t" and a semivowel for "i", and we expect to be able to obtain /š/ from /s y/ by the phonological rule for *palatalization*.[12] Therefore, we define the phonemic pronunciation of "national" to be /n æ s̲ y ɪ n ɪ l/ which may become [n æ š̲ ɪ n ɪ l] phonetically. This

[11] We have previously mapped /t/ to "t" and /y/ to "i" for cases like the word "national." The change did not affect generation performance to any significant extent.

[12] The place of articulation of the alveolar phoneme /s/ often changes to palatal (/š/), upon coarticulation with the phoneme /y/.

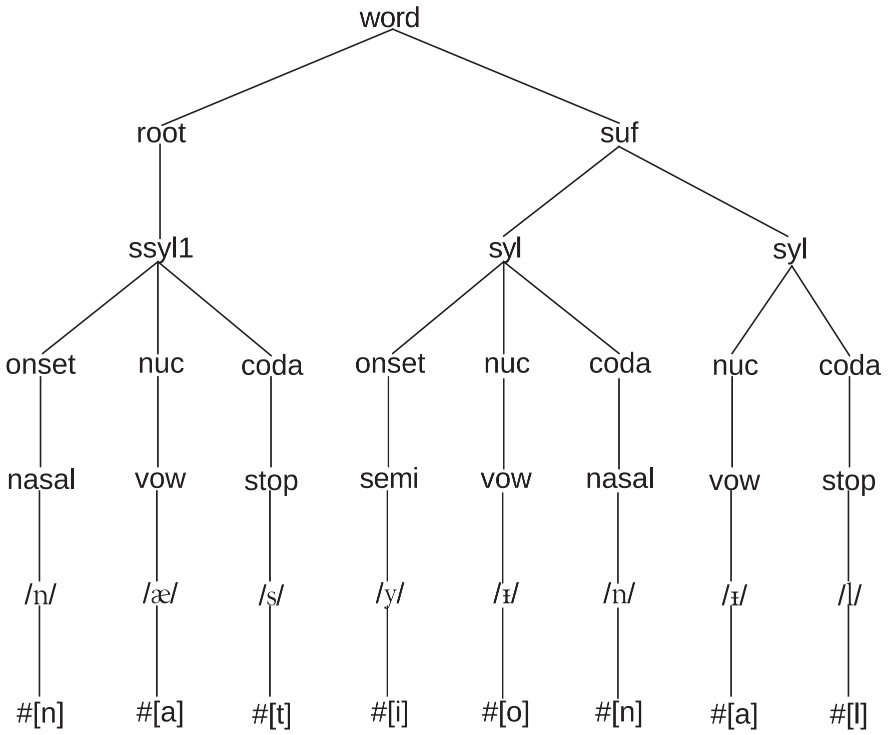


Figure 2-12: Lexical representation for the word "national."

principle enables us to conveniently handle words for which palatalization is optional. The word "issue" (Figure 2-13) can either be pronounced as /ɪ s̲ y u/ (mostly in British English), or /ɪ š̲ u/. If we had used the phoneme /š/ in "issue," then we would be encumbered with a "depalatalization" rule for handling the non-palatalized pronounciation. Likewise, the word "negotiation" can be pronounced as /n i g o s̲ y e š ɨ n/ or /n i g o š̲ y e š ɨ n/, and the word "mature" may be pronounced as /m ɨ t y u r/ or /m ɨ č y u r/.

We have also made an attempt to distinguish the context which calls for a long vowel from that which calls for a short vowel. A long vowel tends to be succeeded by a grapheme containing the letter "e", such as in the /ɑʸ/ in "define" being followed by #[ne] (Figure 2-14) and /e/ in "dedicate" followed by #[te] (Figure 2-3). An effort is made to preserve the context for "defining" (Figure 2-15), by using the terminal

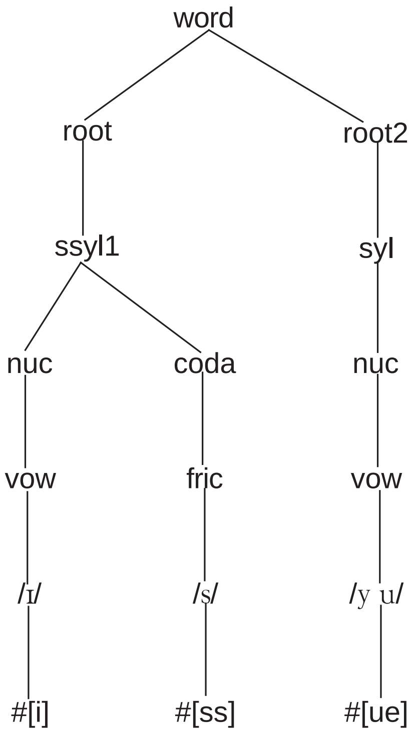


Figure 2-13: Lexical representation for the word “issue.”

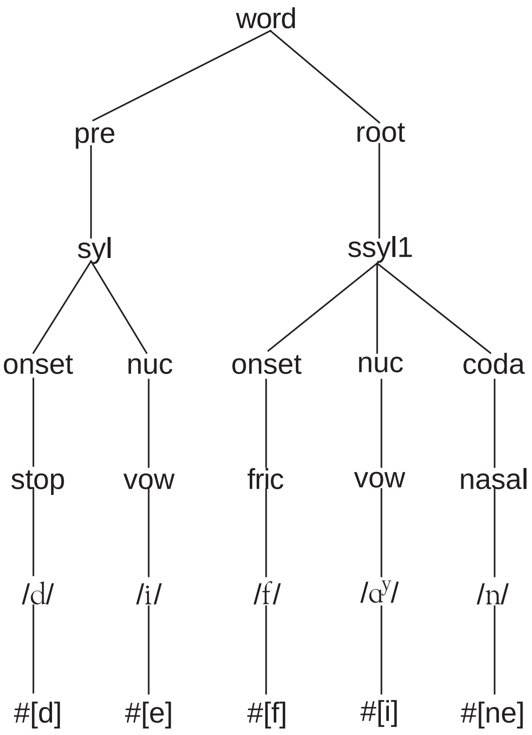


Figure 2-14: Lexical representation for the word "define."

#[n_e]. These *underbar* terminals are restricted to the CODA positions of stressed syllables, and are predicted during the parsing process. Therefore, long vowels tend to transition from left-to-right to the CODA of a grapheme terminal ending with the letter "e" or "underbar" terminal, because the bigram constraint disfavors the advancement to a stressed M-ONSET such as in "definition" (Figure 2-16).

We have also created the unstressed counterparts for the long vowels /i/, /u/ and /o/. The unstressed /i/ is mainly reserved for suffixes, e.g., in "ivory" (Figure 2-17). Another example which also includes the unstressed /u/ is found in "superbly" (Figure 2-18). Finally, an example for the unstressed /o/ is shown in Figure 2-19 for "colorado."

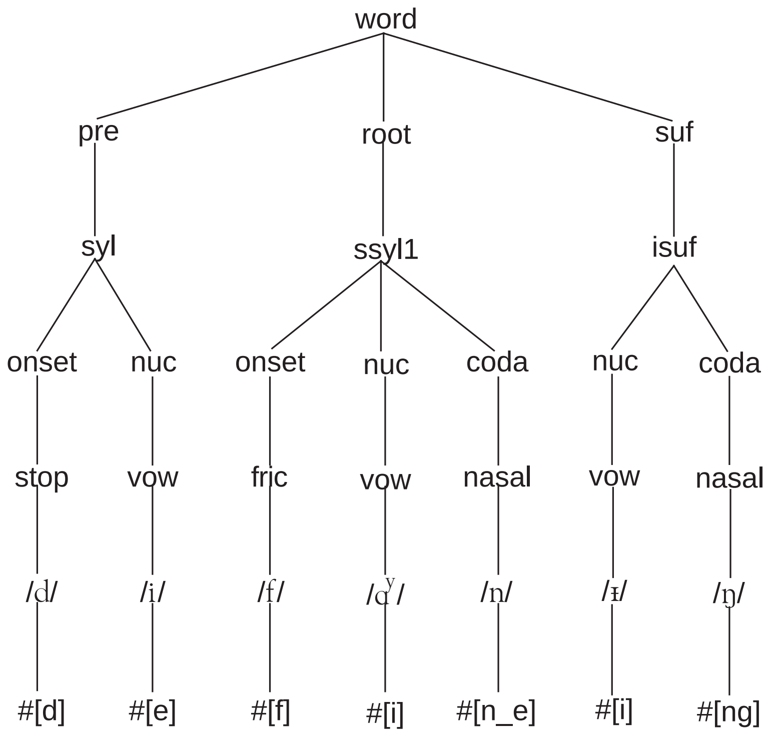


Figure 2-15: Lexical representation for the word "defining."

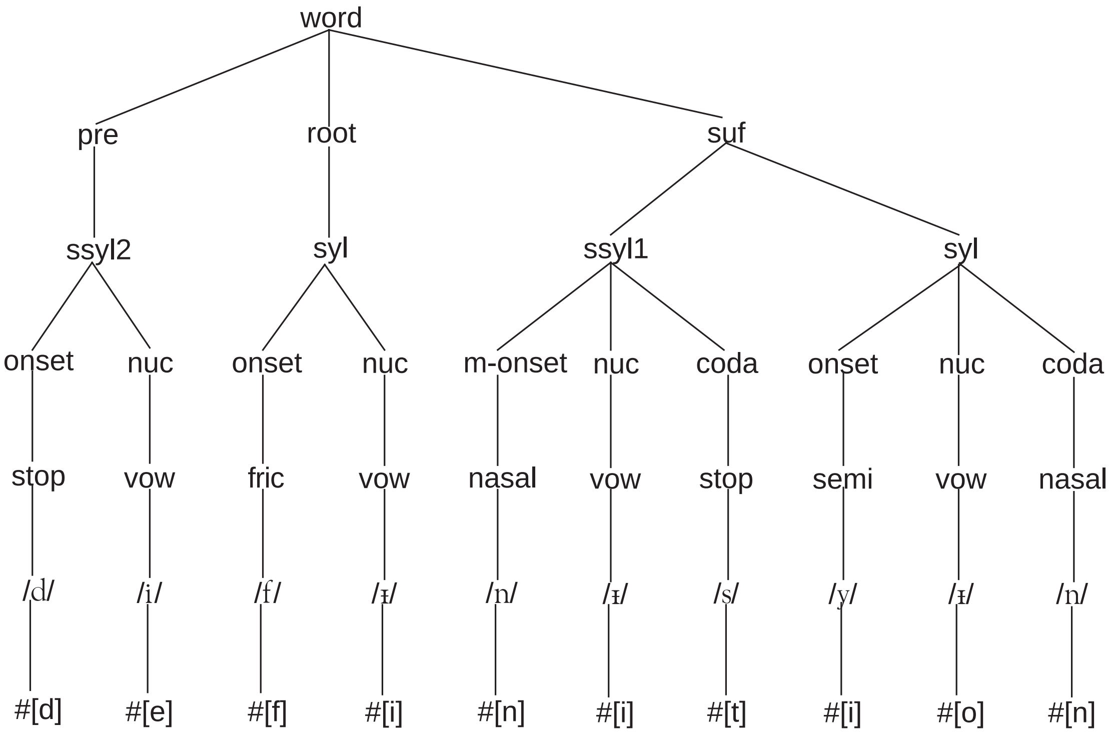


Figure 2-16: Lexical representation for the word “definition.”

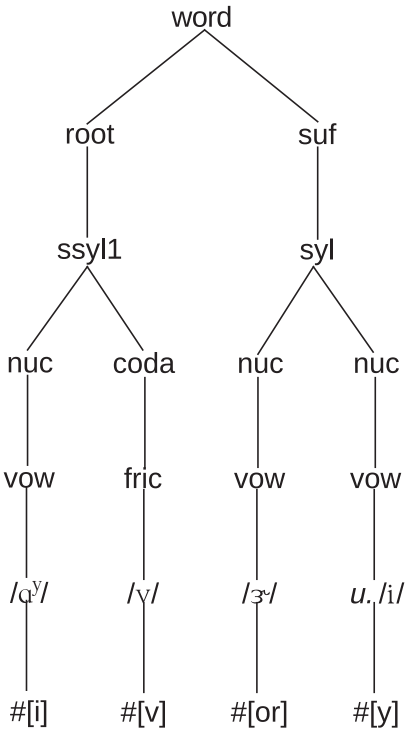


Figure 2-17: Lexical representation for the word "ivory." "*u.* /i/" denotes the unstressed version of /i/.

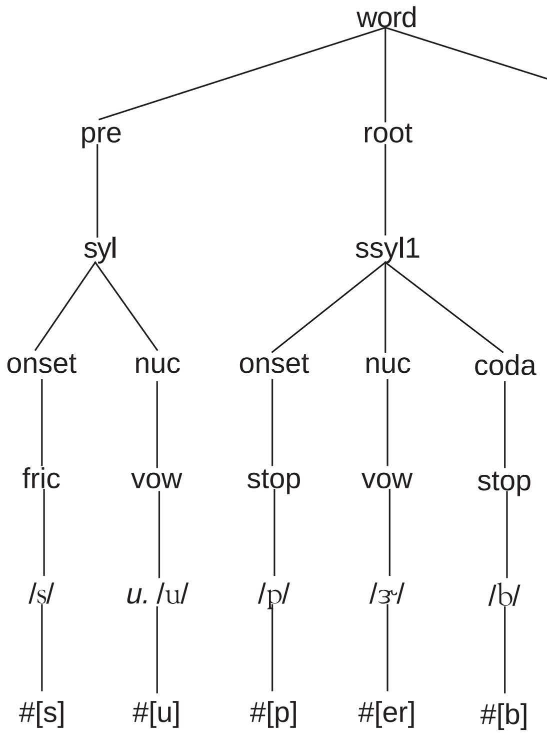


Figure 2-18: Lexical representation for the word "superbly." "*u.* /u/" denotes the unstressed version of /u/.

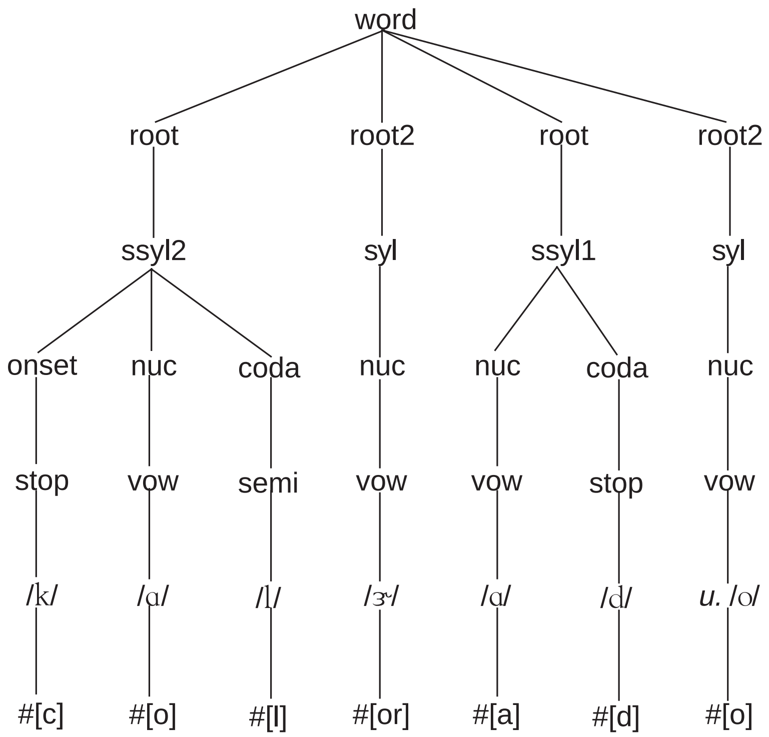


Figure 2-19: Lexical representation for the word "colorado." "*u.* /o/" denotes the unstressed version of /o/.

2.3 Chapter Summary

In this chapter, we present our method of integrating different linguistic knowledge sources to describe English letter-to-sound mappings, with the objective of creating a parsimonious lexical representation. Our design promotes extensive structural sharing among words. The parse trees for words such as "predictable" and "preventable" should be able to share the same structures in the morphology layer, the syllable layer, and all the layers under the prefix (and possibly the suffix).[13] A compact lexical representation with sharing capabilities is potentially applicable and desirable for large-vocabulary speech recognition tasks. It also allows sharing of probabilities which ameliorate the sparse data problem, under the assumptions that sharing takes place among similar distributions only. In this work, the lexical representation is combined with a parsing framework in order to cast the letter-to-sound generation problem as directly symmetric to the sound-to-letter generation problem. We will proceed to describe the parser in the following chapter.

[13] As will be seen in the next chapter, our system generates parse theories from left to right, and the theories are right-branching, i.e., if there are two parse theories which differ only at some intermediate stage, data structures are shared on the left, but not on the right because identical right structures are not merged. Implementation of a merging mechanism should allow structural sharing between the suffixes of "predictable" and "preventable".

Chapter 3

The Parsing Algorithm

The hierarchical lexical representation presented in the previous chapter forms the infrastructure upon which generation is carried out. The approach for generation is one of synthesis-by-analysis in a parsing framework. An input spelling (or pronunciation) is analyzed at all seven linguistic levels in terms of a parse tree, and the generated pronunciation (or spelling) is then derived from the analysis. This chapter describes our generation algorithm. The training procedure is a hybrid of rule-based and data-driven strategies. A small set of context-free rules are written by hand, and are used in accordance with a natural language parser to produce training parse trees from the labelled training corpus — a subset of the 10,000 most frequent words in the Brown corpus [43]. The parse tree produced serves as training data for a probabilistic parsing algorithm based on the "layered bigrams" [74].[1] The straightforward constraints specified by the rules, and other more subtle regularities embodied in the training parse trees, are all converted by the training procedure into a set of probabilities. Therefore, the advantage of the hybrid approach is to trade-off the expensive efforts in providing an elaborate set of letter-to-sound rules from linguistic

[1] The layered bigrams have previously been used to parse sentences in the ATIS domain [66]. In this thesis, a modified version is developed for the subword parsing application.

expertise, with a small set of simple rules augmented by constraints automatically discovered from a body of training data. Probabilities prioritize some constraints over others, and thus elegantly bypass problems with rule interactions and conflicts.[2] The testing procedure uses a monolithic set of probabilities for enforcing constraints in various linguistic levels for *both* letter-to-sound and sound-to-letter generation, which are analogous processes. The non-overlapping test set and the development test set are also subsets of the 10,000 most frequent words in the Brown corpus, and disjoint from the training set. The following is a detailed account of our data preparation processes in generating training parse trees, and the training and testing procedures in the layered bigrams.

3.1 Data Preparation

This section describes the method used to generate training parse trees, which form the rule-based aspect of our hybrid approach. The procedure involves labelling a training corpus, writing a set of context-free rules, and boot-strapping with the natural language parser TINA [73]. TINA has previously been used with the SUMMIT recognizer [97] to parse sentences in the VOYAGER domain [98] for navigation, and the ATIS domain [66] for retrieving air-travel information. The formalism of TINA derives a network from a context-free grammar, and the connecting arcs in the network are associated with probabilities. When a sentence is parsed, a set of parse nodes are created and linked together in a hierarchical parse tree, while traversing explicit paths through the grammar network. Therefore, TINA is also suited for producing hierarchical outputs for words in a parse tree format. In our current application, TINA is only used in a boot-strapping procedure which does not involve trained probabilities. Rather, the parser is constrained by the linguistic labels in the training corpus while it operates on a small set of context-free rules.

[2] The benefits of using a probabilistic framework have been covered in Chapter 1.

D!E=DI+@C?ATE++*D	/dɛdɪketɨd/
H!ER==O	/hɪro/
ST!AND$P?OINT	/stændpɔʸnt/

Table 3.1: Examples of lexical entries in the training corpus.

The training corpus consists of about 8,000 words, which is a subset of the 10,000 most frequent words in the Brown Corpus. Table 3.1 shows several examples of the lexical entries in the training corpus. The spellings are marked with symbols for specifying syllable stress and morphological decomposition.[3] The symbols include markers for prefix (=), ROOT2[4] (==), suffix (+), inflectional suffix (++), compound word ($), moved onset (@), primary stress (!) and secondary stress (?).[5] The linguistic markings in Figure 3.1 for the word "dedicated" are quite straightforward. The prefix is found to the left of the prefix marker (=). In this case, it is "DE," marked with a primary stress. The suffix syllables are found to the right of the suffix syllable markers (+). In between prefix marker and the suffix marker is the root of the word. Similarly, inflectional suffix syllables are found to the right of the inflectional suffix marker (++). The first suffix syllable after the root, "ATE" is marked with secondary stress, and also inherits the letter "C" as the moved onset from the root. The graphemic place-holder [*] is inserted in place of the letter "E" in the inflectional suffix. The word "hero" shows the marking of the ROOT2, "O", and "standpoint" exemplifies the labelling for a compound word.

A small set of context-free rules are written for the TINA parser. These rules serve to incorporate linguistic knowledge in the training parse trees. In Table 3.2, we have included one exemplar rule for every pair of adjacent layers. The first rule in the table states that the WORD category at the top layer can expand to an optional PREFIX category, followed by the ROOT category and an optional SUFFIX category.

[3] The morphological decomposition of the training words are provided by Sheri Hunnicutt.

[4] The second unstressed syllable of the root.

[5] Unstressed syllables are not marked.

1.	word	→	[prefix] root [suffix]
2.	root	→	stressed-syllable [unstressed-syllable]
3.	stressed-syllable	→	[onset] nucleus [coda]
4.	nucleus	→	vowel
5.	nasal	→	(/m/, /n/, /ŋ/)
6.	/m/	→	(“m,” “me,” “mn,” “mb,” “mm,” “mp”)

Table 3.2: Examples of lexical entries in the training corpus.

The second rule states that the morph category ROOT can expand to a stressed syllable (layer 3) followed by an optional unstressed-syllable. The third rule states that a stressed-syllable can expand into an optional ONSET, followed by a NUCLEUS and an optional CODA. The fourth rule requires a NUCLEUS to go to a VOWEL. The fifth rule states that the broad class NASAL (layer 4 in the lexical representation) can go to one of three possible phonemes in layer 5, namely, /m/, /n/ or /ŋ/. Finally, the sixth rule states that the phoneme /m/ can correspond to one of the six graphemes (“m,” “me,” “mn,” “mb,” “mm,” “mp”). In total, about 100 rules are needed for all words in the training corpus to be parsed into training parse trees. An exhaustive listing of the rules is given in Appendix G.

TINA attaches probabilities only to sibling-sibling transitions in the context-free rules. For example, a typical rule such as $\mathit{parent} \rightarrow \mathit{sibling}1\ \mathit{sibling}2$, is bound with two probabilites: $P(\mathit{sibling}1 \mid \mathit{start},\ \mathit{parent})$, the transition probability from the beginning of the rule to the first sibling, under the context of the parent; and $P(\mathit{sibling}2 \mid \mathit{sibling}1, \mathit{parent})$, the transition probability from the first to the second sibling, under the context of the parent. Therefore, the probabilities in TINA only capture constraints between the two levels *within* a context-free rule. This proved to be adequate for parsing sentences in the previous applications, where the probabilities are augmented with a semantic grammar and syntactic features to ensure agreement. Such elements are absent for our current task, and TINA’s formalism led to a great deal of overgeneration while parsing words [59]. We can compensate somewhat by writing a large number of explicit rules to filter the generated hypotheses, but this does not alleviate the heavy computational load in exploring partial theories that would later fail. Therefore, we have reformulated TINA in the paradigm of the "layered bigrams" for our generation tasks. The new framework utilizes *across-rule* constraints in addition to *within-rule* constraints for generation, as will be described in the next section.

3.2 The Training Procedure

The training parse trees are used to train layered bigram probabilities, which constitutes the data-driven aspect of our hybrid approach. The probabilities are the sole parameters used by the parser. Therefore, the training procedure serves to convert the constraints explicitly specified in terms of the context-free rules and linguistic markings, and augment them with more subtle constraints which are automatically discovered from the training parse trees.

The training procedure is geared towards the implementation of a bottom-up, left-to-right parser. We feel that this is a desirable order of processing — bottom-up implies that the more generic categories are predicted based on the more specific categories, which should prevent overgeneration, and avoid generating parse theories that would later fail. Bottom-up parsing is possible in this subword domain because the parse trees have exactly seven layers everywhere.[6] Left-to-right processing is inspired by its success in the development of speech recognition systems.[7] It allows computa-

[6] This was not the case when the layered bigrams were used to parse sentences [74], where some columns in the parse tree have fewer layers than others. The problem is further complicated by the need of a trace mechanism for long-distance movement, feature unification to ensure agreement, and other semantic constraints.

[7] An alternate from of processing, known as the "island-driven" approach, is preferred by some in the speech recognition community. The island-driven approach begins by searching for anchor points in a speech utterance where the confidence level for correct recognition is very high. These anchor points, or "islands of reliability," are then extended in a best-first manner into larger islands, until the islands culminate the speech utterance. This approach was adopted in the Hearsay-II system [48]. An island-driven approach may be better than left-to-right processing because the latter is often

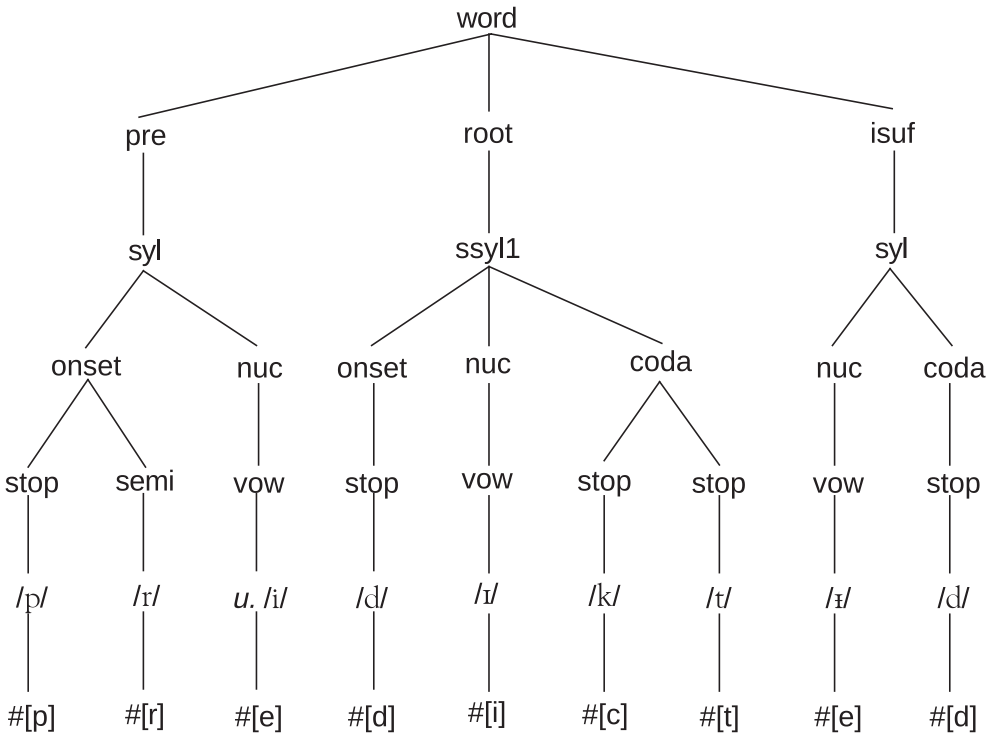


Figure 3-1: A parse tree generated by TINA for the word "predicted." PRE denotes "prefix," ISUF denotes "inflectional suffix," SYL denotes "unstressed syllable," SSYL1 denotes "primary stressed syllable," and NUC denotes "nucleus."

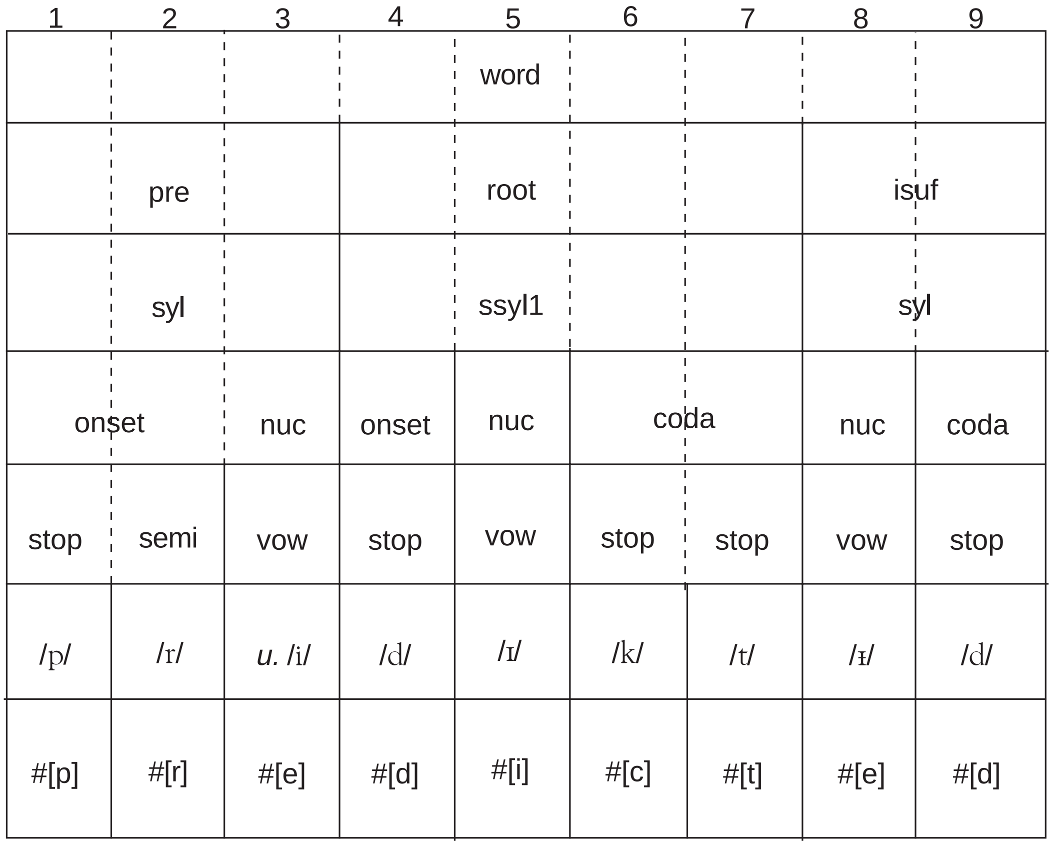
tion to progress as the utterance is received, which is key towards the implementation of real-time systems. Therefore, this bottom-up, left-to-right processing order in the layered bigrams should be an attractive feature for efficient generation in our current tasks, and for potential applications in speech recognition.

In the analysis of a training parse tree, such as the typical example shown in Figure 3-1, a basic 4-tuple is used. It consists of the elements:

1. Right-Sibling (RS): This can be any category in the parse tree.

2. Right-Parent (RP): The category above RS in the parse tree.

forced to deal with difficult, unpromising portions of speech as they occur. Despite this, however, left-to-right processing has remained popular and successful in recent speech recognition systems.



1	2	3	4	5	6	7	8	9
word								
pre			root				isuf	
syl			ssyl1				syl	
onset		nuc	onset	nuc	coda		nuc	coda
stop	semi	vow	stop	vow	stop	stop	vow	stop
/p/	/r/	*u.* /i/	/d/	/ɪ/	/k/	/t/	/ɨ/	/d/
#[p]	#[r]	#[e]	#[d]	#[i]	#[c]	#[t]	#[e]	#[d]

Figure 3-2: The parse generated by TINA for the word "predicted," shown in a parse tree format in the previous figure, but displayed here in layered bigrams format.

3. Left-Sibling (LS): The category to the left of RS in the parse tree.
4. Left-History (LH): The entire column history (consisting of seven categories, one from each layer) to the left of RS.

For illustrative purposes, Figure 3-2 shows the same parse tree as Figure 3-1, but in layered bigrams format. The derivation of a terminal node, such as #[r] in column 2, is as follows:

$$
\begin{aligned}
\textsc{word} &\rightarrow \textsc{pre root isuf}\\
\textsc{pre} &\rightarrow \textsc{syl}\\
\textsc{syl} &\rightarrow \textsc{onset nuc}\\
\textsc{onset} &\rightarrow \textsc{stop semi}\\
\textsc{stop} &\rightarrow /\mathrm{p}/, \quad \textsc{semi} \rightarrow /\mathrm{r}/\\
/\mathrm{p}/ &\rightarrow \#[\mathrm{p}], \quad /\mathrm{r}/ \rightarrow \#[\mathrm{r}]
\end{aligned}
$$

Therefore, if we regard the terminal #[r] as our current node, then the entire first column, i.e. {WORD PRE SYL ONSET STOP /p/ #[p]}, should consitute its left-history (LH). The left-parent (LP) and left-sibling (LS) are respectively defined as the phoneme /p/ and the letter #[p] in column 1, while the right-parent is defined as the phoneme /r/ in column 2. Generally, the left and right parents may or may not be identical. Notice that in this example the LP is different from the RP, because this 4-tuple is derived from *two* different context-free rules: /p/→#[p], and /r/→#[r].

As another example, consider the category NUC in column 5 in Figure 3-2. The steps involved in its derivation are:

$$
\begin{aligned}
\textsc{word} &\rightarrow \textsc{pre root isuf}\\
\textsc{root} &\rightarrow \textsc{ssyl}1\\
\textsc{ssyl}1 &\rightarrow \textsc{onset nuc coda}
\end{aligned}
$$

It is obvious that column 4, i.e. {WORD ROOT SSYL1 ONSET}, should be the LH, LS is ONSET, LP is SSYL1 and RP is SAME. Notice that in this case, the LP is identical to RP, because this 4-tuple is derived from a single context-free rule, namely, SSYL1→[ONSET] nucleus [CODA]. In other words, a 4-tuple encodes "within-rule" constraints in a derivation if LP and RP are equivalent. Otherwise, if LP and RP are different, such as in the example above, the 4-tuple corresponds to "across-rule" constraints.

The design of our probabilistic formalism evolves around the bottom-up, left-to-right parsing order. There are two independence assumptions made:

1. In left-to-right prediction, the probability of predicting the current category is independent of all context except for its immediate left-history. We feel that the use of the *entire* left-history integrates context from previous derivations, which envelopes all the upper layers in the hierarchical structure. This is much more constraining than the original TINA formalism which involves only a single derivation. However, the context further left is shrouded. We choose to keep the conditioning context simple to ease parsing, and to avoid serious sparse data problems due to over-specificity.

2. In bottom-up prediction, the predicted parent category is also conditioned on its immediate left-history, as well as the child category. The prediction probability is assumed independent of the context further beneath the child. The intent is, once again, to promote sharing of training data.

The training probabilities are computed by tallying counts and then normalizing them by the total counts. Each word in the lexicon is counted as a single occurrence.[8] The set of training probabilities includes:

1. start terminal unigram $P_{start\ terminal\ unigram}(start\ terminal)$ — this is the unigram probability over all the terminals that can start a word. In the letter-to-sound generation case, the start terminal is a grapheme, e.g., the letter #[p]

[8] Another possibility is to take the word frequencies into account.

starts the word "predict," and the grapheme #[ph] starts the word "philosophy."

2. start column prediction probability $Pr(RP \mid RS,\ LH\ =\ \textsc{start})$ — this is the bottom-up prediction probability given that we are at the start column of the word. The "start column probability" is the product of the start terminal unigram and all the bottom-up prediction probabilities in the start column, i.e.,

$$\begin{aligned} P_{start\ column} = \ & P_{start\ terminal\ unigram}(start\ terminal) * \\ & \textstyle\prod_{r=2}^{r=7} P(RP_{(r-1)} \mid RS_r\ ,\ LH = \textsc{start}) \end{aligned} \tag{3.1}$$

where r is the row-index, $r = 7, 6, 5, ...1$,
RS_r is the right-sibling at row r,
(RS_1 = WORD, RS_7=start terminal)
RP_r is the right-parent at row r.

3. column advance probability $Pr(RS\ =\ next\,terminal \mid LH\ =\ current\,column)$ — this is the bigram probability over all the terminals than can follow the current column.[9] The next terminal may be an END node.

4. column prediction probability $Pr(RP \mid RS,\ LH)$ — this is the bottom-up prediction probability conditioned on the left-history and the current (right-sibling) category. The layered bigrams have been modified to be driven entirely bottom-up so that the "within-rule" statistics and "across-rule" statistics are merged. The bottom-up prediction probability $Pr(RP \mid RS, LH)$ makes a prediction using the entire left-history as its left context. The "column probability" is the product of the column advance probability and all the bottom-up prediction

[9] The use of pseudo-diphthongs in this case is favorable. Pseudo-diphthongs, such as /ol/, combine two phonemes into one. Therefore, the advancement probability for pseudo-diphthongs is in reality capturing trigram constraints.

probabilities in the current column which we are trying to construct, i.e.,

$$\begin{aligned} P_{column} = \; & Pr(RS \; = \; \textit{current terminal} \mid LH) \, * \\ & \textstyle\prod_{r=7}^{r_i} P(RP_{(r-1)} \mid RS_r \; , \; LH) \end{aligned} \qquad (3.2)$$

where r is the row-index, $r = 7, 6, 5, ...1,$

RS_r is the right-sibling at row r,

$(RS_1 = \textsc{word}, \; RS_7 = \textit{terminal layer})$

RP_r is the right-parent at row r

$RP_{r_i - 1} = \textsc{same}.$

Notice that we stop accumulating column prediction probabilities once we reach $RP = \textsc{same}$. This is because from then on the right-history merges with structures which are already in place in the left-history due to previous derivations from the context-free rules.

3.3 Testing Procedure

Bi-directional generation during testing is achieved by constructing a layered parse structure based on the *input* spelling (or pronunciation), and the *output* pronunciation (or spelling) is derived from the parse tree.

In letter-to-sound generation, the procedure commences by selecting the appropriate start terminal categories,[10] and generating start columns *bottom-up* for each of the terminals. Sound-to-letter generation begins by predicting the start column *bottom-up* based on the first phoneme, and this partial history determines the terminal categories that can go beneath it. A complete start column is then generated

[10] These are the terminal categories with the same letter sequence as the beginning of the input word spelling. For example, in parsing the spelling "eight" from left-to-right, the terminals #[e], #[ei] and #[eigh] (which maps to /e/ in "eight") are all considered.

for each of the terminal categories. Every start column records its own start column probability ($P_{start\ column}$). The columns are pushed on a stack to become partial parse theories while the start column probabilities become the stack scores.

At each iteration of the generation algorithm, the stack is sorted, and the partial theory with the highest stack score is popped off the stack. This theory is then advanced *left-to-right*. The advancement procedure checks to see if the previous column of the partial theory (i.e. the entire LH) is valid top-down, and if it can reach any of the possible terminal categories (RS) that follow.[11] If either one of these conditions is not satisfied, the partial theory is eliminated. Otherwise, the partial theory is advanced to the next possible terminals, and each of these terminals will produce its own history *bottom-up*. In essence, the original theory popped off the stack will spawn off a handful of new partial theories, each constituting the original theory extended by a new column. The stack scores for these new partial theories are computed using the appropriate left-to-right advancement probabilities ($Pr(RS \mid LH)$) and bottom-up prediction probabilities for generating a column (P_{column}). The method of computation will be described in detail in the next section. The new partial theories are then pushed back onto the stack and the process repeats. The iteration continues until one or more *complete* theories are popped off the stack. A *complete* theory is one which can account for the entire input, i.e., it has the complete word spelling in its terminal layer for letter-to-sound generation, or the complete phonemic sequence in its pre-terminal layer for sound-to-letter generation. In addition, a complete theory also contains an END node in its last column.

It can be seen that the layered bigrams algorithm applies very *local* probabilistic constraints between adjacent layers in the lexical representation. Performing this throughout the ordered layers in the hierarchy results in the *implicit* enforcement of some *long-distance* "bigram" constraints. Specifically, if we examine closely some of

[11] The possible right-siblings (RS) include the *underbar* terminals which are predicted based on the left-history (LH). A list of the *underbar* terminals can be found in Appendix F.

the generated phoneme sequences provided in the next chapter, we should see that bigram constraints in the morph layer and syllable layer have propagated downwards to the phoneme layer. Should constraints *beyond* the bigram be desired, *explicit* enforcement will be required through the use of filters. For example, the syllable stress layer permits a reduced syllable SYL to follow another reduced syllable. If only bigram constraints are applied during parsing, it will not be surprising to obtain a parse output where all the syllables in a multi-syllable word are reduced, albeit with low probability. There are also bisyllabic words with a PREFIX-ROOT morphology, where the nouns often have a SYL-SSYL1 stress pattern, and the verbs have a SSYL1-SYL pattern (consider "permit," "record," etc.). Furthermore, there are the "stress-affecting suffixes" such as "-ation," which tend to alter the stress contour of a word in a predictable way (consider "combine" vs. "combination"). Stress filters can be used for these cases to eliminate any partial (or complete) theories which have illegitimate stress patterns. Similarly, it is possible to use morph filters to eliminate theories which violate morphotactic constraints. Apart from allowing additional constraints to be enforced, the flexibility of the layered bigrams algorithm also allows us to relax some constraints. This can be achieved by a "backoff" mechanism on the column advancement probabilities to expiate sparse training data problems, and increase the coverage of the parser. A robust parsing strategy for the layered bigrams will be presented in Chapter 6.

3.4 An Efficient Search Algorithm

Our parsing procedure employs an *ordered, best-first* search algorithm. The ordering is governed by an evaluation function for computing the stack score. The choice of this evaluation function is important for maximizing search efficiency and guaranteeing admissibility of the search, as will be explained in the following.

Define the function $f(c)$ at any column c in the layered bigrams parse tree to be the sum of the actual log-likelihood of an optimal path from the START column to c, denoted by $g(c)$, and the log-likelihood of an optimal path from n to an END node, denoted by $h(n)$, i.e.

$$f(c) = g(c) + h(c) \tag{3.3}$$

The goal of the search is to find the path with the maximum log-likelihood. We can choose an evaluation function $\hat{f}(c)$ to be an estimate of $f(c)$ which is given by:

$$\hat{f}(c) = g(c) + \hat{h}(c) \tag{3.4}$$

where $\hat{h}(c)$ is the estimate of $h(c)$.

An obvious choice of $g(c)$ is the highest log-likelihood from s to c found so far by the algorithm. As for $\hat{h}(n)$, the choice is quite influential on search efficiency. If $\hat{h}(n)$=0, is used, then shorter partial paths will generally have better stack scores than longer partial paths because they accumulate fewer log-likelihoods. The search will then be thrust towards extending short partial paths and end up with a very large stack, i.e., a *uniform search* will result. Therefore, it is important to have an estimate of the future score of the path from the current column to the END. It can be shown [60] that if $\hat{h} \geq h$, then the search algorithm is *admissible*, i.e., the first complete path delivered by the search will be the path with maximum log-likelihood, but admissibility is not guaranteed otherwise. If $\hat{h}$ equals the tightest upper bound on h, then the search will become the optimal A^* search, and the number of extensions required to obtain the best path is kept to a minimum.

Therefore, computing a look-ahead score which is a tight upper bound of the future score would guarantee optimality and admissibility of the search. However, this computation may also be costly, or even problematic in real-time applications when we do not know where a path will end. Since efficiency can often be gained at the expense of forsaking admissibility, we are currently using an evaluation function which invokes a score normalization mechanism. This mechanism aims at generating stack scores within a certain numeric range, and thus strives to achieve a fair comparison on the goodness of a partial path between the shorter partial paths and the longer ones. Scoring normalization may be accomplished by an *additive* correction factor in some cases, and a *multiplicative* correction factor in others. In our implementation, we use a "fading" scheme as shown in the equation:

$$\hat{f}(c) = \alpha \hat{f}(c') + (1 - \alpha) p(c', c) \tag{3.5}$$

where $\hat{f}(c)$ is the stack score from the START column to the current column c,
c' is the column preceding c in the parse tree,
$p(c', c)$ is the log-likelihood associated with extending the parse tree
from c' to c, and
α is some fading factor $(0 < \alpha < 1)$

The idea is to have the stack score carry short term memory, where the new column always contributes towards a certain portion of the stack score (according to the pre-set weight of $\alpha = 0.95$), while the remaining portion associated with the past gradually fades away, so that the distant past contributes less to the stack score than the recent history, and the score tends to remain quite stable over time. The outcome of this search is that the ordering tends to place together parse theories with similar distant columns and different recent columns.

If multiple hypotheses are desired, the algorithm can terminate after a desired number of complete hypotheses have been popped off the stack. In addition, a limit is set on the maximum number of theories (partial and complete) popped off the stack. The complete theories are subsequently re-ranked according to their actual parse score. The idea is to first use the "fading" stack criterion as an inexpensive means to obtain a handful of plausible complete hypotheses, and follow up with a more careful means of ranking (with no fading) in order to find the "best" theory. Though our search is inadmissible, we are able to obtain *multiple* hypotheses inexpensively.

3.5 Chapter Summary

This chapter explains our generation algorithm, based on a hybrid approach. The approach has a rule-based part, where a set of context-free rules are hand-crafted for generating training parse trees. The approach also has a data-driven part, where the training parse trees are used to train up the probabilities in a probabilistic parser, christened the "layered bigrams algorithm." The training algorithm serves to encode the constraints specified in the rules into a set of probabilities. The probabilities also augment the rules with other constraints automatically discovered from the training parse trees. The testing algorithm adopts a best-first search strategy, and produces a complete parse tree based on an input spelling/pronunciation, and from this the output pronunciation/spelling is derived. In the next chapter, we will report on the generation performance of this parser.

Chapter 4

Experimental Results

This chapter reports on the performance of our parser for both letter-to-sound and sound-to-letter generation. Our experimental corpus consists of the 10,000 most frequent words appearing in the Brown Corpus [43], and each lexical entry contains a spelling[1] and a single phoneme string as its pronunciation. The words are ordered alphabetically, and entries which are marked as either function words (e.g. "A," "AM," "BY," and "HAVE") or abbreviations (e.g. "AUG," "CORP," "ETC.") are discarded.[2] Every tenth word is set aside as a future test set, and every tenth word of the remaining set is aside as the development test set. The rest of the words (about 8,000 in total) are used for training. The results given in this chapter are based on the development test set only, as slight improvements and a robust parsing mechanism will ensue. The best configuration of our system, based on the attainment of the highest performance and broadest coverage on the development test set, is ultimately tested on the *real test set*, and these results will be reported in Chapter 6.

The parser was set to terminate after obtaining up to a maximum of 30 complete parse theories, or after the maximum number of theories (partial and complete)

[1] The word spellings are marked to indicate stress and morphological decomposition.

[2] There are 97 function words and 32 abbreviations in all. Function words are omitted because they tend to have different letter-to-sound mappings from other English words [3].

popped off the stack reaches 330, whichever happens first. These numbers are empirically chosen as a limit on the depth of the search. The thirty hypotheses are then re-ranked according to their actual parse score, and the performance accuracies reported below are based on the new set of rankings.

In the following we will describe our evaluation criteria, report on the results for letter-to-sound and sound-to-letter generation, and finally provide an analysis of errors for the generation tasks.

4.1 Evaluation Criteria

The two criteria which we use for evaluating letter-to-sound/sound-to-letter generation accuracies are similar to those used in the other systems reported previously.

1. Word accuracy — In the case of letter-to-sound generation, one can perform a match between a generated phoneme string and the reference phoneme string from the lexical entry of the word. Our experimental corpus provides only a *single* reference pronunciation per word. Generation is correct if there are no discrepancies between the two phoneme sequences. In the case of sound-to-letter generation, a similar match is performed between the two letter sequences. This is a strict evaluation criterion which does not permit alternate pronunciations for words, as any deviation from the reference string is regarded as an error.

2. Letter/Phoneme accuracy — In order to indicate the extent to which a generated spelling or pronunciation is correct, phoneme accuracy should be a good evaluation criterion to use for letter-to-sound generation, while letter accuracy should be used for sound-to-letter generation. The generated string is aligned with the "correct" string using a dynamic programming algorithm, which selects the alignment with the minimum number of insertion, deletion and substitution operations necessary to map the generated string to the reference string. The accuracy is computed by subtracting the sum of the insertion (I), deletion (D)

> and substitution (S) error rates from 100%, i.e.
>
> $$accuracy = 100\% - (I + D + S)\% \tag{4.1}$$
>
> This evaluation criterion is the one adopted by NIST for measuring the performance of speech recognition systems.

The letter/phoneme accuracy evaluation criterion assumes that all discrepancies between the reference and generated strings have equal costs. This may not be a fair assumption because often a word has alternative pronunciations which are not provided by the lexicon. Moreover, certain confusions tend to be more acceptable than others. Vowel-vowel confusions in a reduced syllable, or confusions involving few differences in distinctive features are often tolerable. For example, one would probably allow the pronunciation for "proceed" to be transcribed as /p r o s i d/ as well as /p r ʌ s i d/, but this /o/ and /ʌ/ confusion is unacceptable for the stressed vowels in "boat" and "but". Therefore a better method of evaluation is to elicit the opinions from human subjects. However, since this thesis does not emphasize performance comparison with other systems,[3] we have not undertaken the task of conducting human evaluation.

It should also be noted that although there are quite a few existing spelling-to-pronunciation systems, thus far there are no standardized data sets or evaluation methods employed. As described in the section on previous work (Chapter 1), evaluation criteria for letter/sound conversion that have previously been used include word accuracy (which may be based on human judgement), spelling accuracy per letter, pronunciation accuracy per phoneme and pronunciation accuracy per letter. Errors in the generated stress pattern and/or phoneme insertion errors may be neglected

[3] A careful study comparing the performance of eight name-pronunciation system can be found in [30].

in some cases. However, the phoneme accuracy measurement which we use above includes insertion penalties. To a certain extent, stress errors are also accounted for, since some of our vowel phonemes are stress-loaded, i.e. we distinguish between their stressed and unstressed realizations. In measuring pronunciation accuracy per letter, silent letters are regarded as mapping to a /null/ phoneme. We believe that pronunciation accuracy per letter would generally be higher than per phoneme, because there are on average more letters than phonemes per word. In order to substantiate this claim, we tested on our training set, and measured the performance using *both* pronunciation accuracy per phoneme and per letter, based on the alignment provided by the training parse trees. Our results show that using the *per letter* measurement led to approximately 10% reduction in the quoted error rate. It should be kept in mind that throughout the thesis, we will be quoting *per phoneme* results.

4.2 Results of Letter-to-Sound Generation

In letter-to-sound generation, about 6% of the development test set was nonparsable. This set consists of compound words, proper names, and other words that failed due to sparse data problems. Results for the parsable portion of the test set are shown in Table 4.1. The 69.3% word accuracy corresponds to a phoneme accuracy of 91.7%, where an insertion rate of 1.2% has been taken into account, in addition to the substitution and deletion errors. Our phoneme accuracy lies within the low 90's percentage range of the automatic letter-to-sound generation systems described in Chapter 1. The word accuracies of the rule-based approaches, which is typically in the mid 80 percentage range, is considerably higher than our top-choice word accuracy, but comparable to our N-best accuracy with $N = 5$. This may suggest that we can seek performance improvement by means of better search procedures. Alternatively, we can try to improve performance by using more contextual information during parsing, or devise post-processes to select among the top few generated outputs.

Accuracy		top choice correct	top 5 correct	top 10 correct
train	word	77.3%	93.7%	95.7%
	phoneme	94.2%	–	–
test	word	69.3%	86.2%	87.9%
	phoneme	91.7%	–	–

Table 4.1: Letter-to-sound generation experiments: Word and phoneme accuracies for training and testing data. Nonparsable words are excluded.

This will be further addressed in Chapter 8.

Figure 4-1 is a plot of cumulative percent correct of whole word theories as a function of the N-best depth for the development test set. Although 30 complete theories were generated for each word, no correct theories occur beyond N =18 after re-sorting. Performance reaches an asymptotic value just beyond 89%.

4.3 Results on Sound-to-Letter Generation

In sound-to-letter generation, about 4% of the test set was nonparsable. Results for the parsable words are shown in Table 4.2; top-choice word accuracy for sound-to-letter is about 52%. This corresponds to a letter accuracy of 88.6%, with an insertion error rate of 2.5% taken into account in addition to substitution and deletion errors. Recall from Chapter 1 that sound-to-letter generation systems using disjoint training and testing data sets typically report word accuracies in the 20 percent range. Using this as a rough estimate, our system compares favorably with the other systems.

Figure 4-2 is a plot of the cumulative percent correct (in sound-to-letter generation) of whole word theories as a function of N-best depth of the test set. The asymptote of the graph shows that the first 30 complete theories generated by the parser contain a correct theory for about 83% of the test words. Within this pool, re-sorting using the actual parse score has placed the correct theory within the top 10 choices for about 81% of the cases, while the remaining 2% have their correct theories ranked

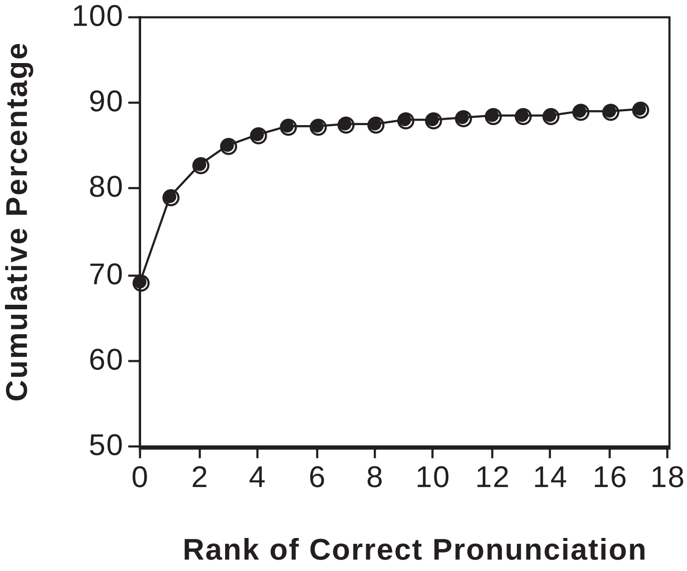


Figure 4-1: Letter-to-sound generation experiments: Percent correct whole-word theories as a function of N-best depth for the test set.

between $N = 10$ and $N = 30$. Re-sorting seems to be less effective in the sound-to-letter case, presumably because many more "promising" theories can be generated than for letter-to-sound. For example, the generated spellings from the pronunciation of "connector" i.e., the phoneme string /k ɪ n ɛ k t ɝ/, include: "conecter," "conector," "connecter," "connector," "conectar," "conectyr," "conectur," "connectyr," "connectur," "conectter," "connectter" and "cannecter." A possible reason for this is the ambiguity in phoneme-to-letter mapping, e.g., the phoneme /ɝ/ above is mapped to "er," "or," "ar," "yr" and "ur". Another reason is that geminant letters are often mapped to the same (consonantal) phoneme, e.g., the phoneme /n/ above can be mapped to "n" or "nn." Many of these hypotheses can be rejected with the availability of a large lexicon of legitimate English spellings.

Accuracy		top choice	top 5	top 10
		correct	correct	correct
train	word	58.8%	85.0%	89.3%
	letter	90.6%	–	–
test	word	51.9%	77.0%	81.1%
	letter	88.6%	–	–

Table 4.2: Sound-to-letter generation experiments: Word and letter accuracy for training and testing data

4.4 Error Analyses

Both of the cumulative plots shown above reach an asymptotic value well below 100%. Computation for the cumulative percentages include words for which the generated output is "correct," as well as other words for which the "correct" theory does not surface with top-rank, but is among the N-best. In some cases, different parse trees in the N-best pool may give the same output spelling/pronunciation, but with different higher level linguistic analyses. Therefore, another possible method for N-best rescoring is to sum the independent probabilities of the different parse theories with the identical phonemes, and re-rank the generated pronunciations.

In order to retrieve the "correct" hypotheses from the N-best pool, we can perhaps adopt a better stack criterion to target admissibility and curb search errors. Alternatively, we can eliminate systematic errors and refine generation outputs by post-processing with additional contextual information. A pilot experiment is conducted along these lines, using a technique known as *transformational error-driven learning* [9]. The study will be described in Chapter 8.

The words that belong to the portion of the test set lying above the asymptote appear intractable — a correct pronunciation/spelling did not emerge as one of the 30 complete theories. We have grossly classified the errors into four categories: (1) Generated pronunciations that have subtle deviations from the reference strings. (2) Unusual pronunciations due to influences from foreign languages. (3) Generated pro-

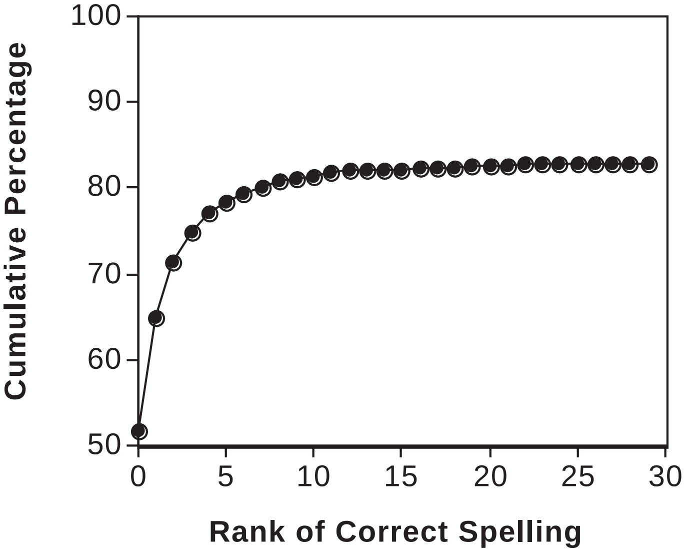


Figure 4-2: Sound-to-letter generation experiments: Percent correct whole-word theories as a function of N-best depth for the test set

nunciations which agree with the regularity of English letter-phoneme mappings, but were nevertheless incorrect. (4) Errors attributable to sparse data problems. Some examples are shown in Table 4.3. It is interesting to note that much overlap exists between the set of problematic words in letter-to-sound and sound-to-letter generation. This suggests that improvements made in one generative direction should carry over to the opposite direction as well.

Certain pronunciation errors, such as the generated pronunciation for "acquiring," /ᵻ k w ɑʸ r ᵻ ŋ/, may be considered by some as correct. Likewise are other examples such as /p æ s y ᵻ n e t/ generated from "passionate" instead of /p æ s y ᵻ n ᵻ t/,[4] /k ᵻ r t u n/ for "cartoon" instead of /k ɑ r t u n/, and /p i p l̩/ for "people" instead of /p i p ᵻ l/. These cases can perhaps be rectified if alternate "correct" pronunciations were available. Spelling errors that are near misses — words like "viscossity" (instead of the correct spelling "viscosity"), "abundent" (instead of "abundant"), "id-

[4] Recall from Chapter 2 that we are using the "underlying phonemic" form, and as a result the /š/ phoneme is transcribed as /s y/ in "passionate".

Category	correct spelling	generated spelling	generated pronunciation	correct pronunciation
(1) Subtle	acquiring	equiring	/ɨkwɑ^{y}rɨŋ/	/ɨkwɑ^{y}ɝɨŋ/
	balance	balence	*correct*	/bælɨns/
	launch	lawnch	*correct*	/lɔnč/
	pronounced	pronounst	/prɨnɑ^{w}nst/	/pronɑ^{w}nst/
(2) Unusual	champagne	shampain	/čæmpɨgni/	/šæmpen/
	debris	dibree	/dibrɨs/	/dɨbri/
(3) Regular	basis	*correct*	/bæsɨs/	/besɨs/
	elite	aleat	/ɨlɑ^{y}t/	/ɨlit/
	violence	viallence	*correct*	/vɑ^{y}ɨlɨns/
	viscosity	viscossity	/vɪskosɨti/	/vɪskɑsɨti/
(4) Sparse	braque	brack	/brækwi/	/bræk/

Table 4.3: Some examples of generation errors.

iological" (instead of "ideological"), or names like "ilynoy" (instead of "illinois") or "claten" (instead of "clayton"), can perhaps be salvaged by spell-checking or verification.

4.5 Data Partitioning

One may presume that our method of partitioning the experimental data, which has test words evenly distributed across the lexicon, would lead to higher generation performance than other methods of partitioning.[5] In order to address this question, we ran a series of four experiments using only the training set, from which we held out one tenth of the words for testing. In one experiment, the test words were extracted as one of every ten words in the original training set, so that an "even" distribution results. The word accuracy of this test set on spelling-to-pronunciation generation was 64.4%. In the other three experiments, the test words were extracted randomly

[5] For example, if we were to train on the first half of the corpus and test on the second half, then we would probably have difficulty parsing many more words, e.g., all the words that begin with the letter "Z," since we have not observed parses which start with the grapheme terminal #[z] in the training data.

from the original training set. The mean word accuracy obtained was 65.3% with a variance of 0.8. Therefore, selecting test words evenly from the lexicon does not raise generation performance.

4.6 Chapter Summary

This chapter presents the performance of our parser for bi-directional generation on the development test data. Results are reported based on word accuracies and phoneme/letter accuracies. These are strict criteria which demand an exact match between the generated string and the *single* "correct" reference string. Competitive results are obtained for both generation tasks. Illustrative examples of generation errors are provided.

Chapter 5

Evaluating the Hierarchy

In the previous chapters we have presented a system which is capable of automatic bi-directional letter-sound generation. The design adopts a probabilistic parsing approach which incorporates a hierarchy of linguistic knowledge for capturing English orthographic-phonological regularities. This framework has enabled us to formulate letter-to-sound generation as a directly symmetric problem to sound-to-letter generation, thereby achieving reversibility in a single system. We believe that the higher level linguistic knowledge incorporated in the hierarchy is important for our generation tasks. Consequently, we would like to empirically assess:

1. the relative contribution of the different linguistic layers towards generation accuracy, and

2. the relative merits of the overall hierarchical design.

This chapter describes two studies—the first investigates the importance of each layer in the *hierarchical* framework, by observing how performance is affected by omitting the layer. The second compares the hierarchical system with an alternative approach which does not have access to higher level linguistic knowledge. We will refer to it as the *non-linguistic* approach. These studies are conducted for letter-to-sound generation only. The implications of this study are expected to carry over to sound-to-letter generation by virtue of the symmetry between the two tasks.

5.1 Investigations on the Hierarchy

In order to explore the relative contribution of each linguistic level in the generation task (our current focus being letter-to-sound generation), we have conducted a series of experiments whereby an increasing amount of linguistic knowledge (quantified in terms of the number of layers in the hierarchy) is omitted from the training parse trees. The system is re-trained on the training set and re-tested on the development test set for each reduced configuration. Four measurements are recorded for each experiment:

1. Top-choice word accuracy on the development test set, where a word is considered correct when there is an exact match between the generated phoneme string and the *single* pronunciation provided by the lexical entry.

2. Perplexity, i.e., the average number of distinct possibilities for the next grapheme, as predicted from the current grapheme, which may be interpreted as the geometric mean of the possible choices for the next grapheme. It is computed as:

$$PP = exp\{\frac{1}{n} * ln(P_T)\} \tag{5.1}$$

where PP is the perplexity,
T is the parse theory,
P_T is the probability of the parse theory T, which is the product of all the bottom-up prediction and left-to-right advancement probabilities involved, and
n is the number of columns in the parse theory, including the END column.

3. Coverage of the test set, obtained by subtracting the percentage of nonparsable words from 100%. Nonparsable words are those for which no complete parse is generated.

4. The number of system parameters.

The top-choice word accuracy and perplexity reflect the amount of constraint provided by the hierarchical representation, while coverage exhibits the extent to which the parser can share training data across different layers in the hierarchy, so that it can generalize and process previously unseen structures. The number of system parameters is a measurement from which one can observe the parsimony of the hierarchical framework in capturing and describing English orthographic-phonological regularities. It also provides some indication of the computational load required by the configuration of the system.

With the omission of each linguistic level, we expect to see two antagonistic effects on generation accuracy — the diminishing use of linguistic knowledge decreases the amount of constraint provided for generation, which should cause degradation in performance. On the other hand, relaxing constraints brings about more sharing of training data across levels. This should help alleviate the sparse data problem and enhance wider coverage, which may potentially contribute to performance improvement.

5.1.1 Results

The experimental results on investigating the hierarchy are plotted in the Figures 5-1 to 5-4. The different reduced configurations include:

1. no omission

2. omitting the morphology layer,

3. omitting the stress layer,

4. omitting the broad class layer,
5. omitting the morphology and broad class layers,
6. omitting the stress and broad class layers,
7. omitting the morphology, stress and broad class layers, and
8. omitting the morphology, stress, syllable and broad class layers.

The system uses 26 letters, 1 graphemic place-holder and 52 phonemes (including several unstressed vowels and pseudo diphthongs such as /ɔr/). Word accuracy refers to the percentage of the test set for which a correct pronunciation is generated from the word spelling. Nonparsable words are counted as errors. This is different from the word accuracy reported earlier in the previous chapter, which is computed based on the parsable fraction of the test set. The number of system parameters in each case is rounded to the nearest hundred.

The advantages of using higher level linguistic knowledge for letter-to-sound generation can be gleaned from the Figures 5-1, 5-2, 5-3 and 5-4. Each layer in the hierarchical representation embodies one type of linguistic knowledge, and for every layer omitted from the representation, linguistic constraints are usually lost, manifested as a lower generation accuracy, higher perplexity and greater coverage. Fewer layers also require fewer training parameters.

Such phenomena are generally true except for the case of omitting the layer of broad classes (layer 5), which seems to introduce *additional* constraints, thus giving a higher generation accuracy, lower perplexity and lower coverage. This can be understood by realizing that broad classes can be predicted from phonemes with certainty,[1] and therefore the broad class layer provides no additional linguistic constraint. The

[1] These unity probabilities are also counted as system parameters. Broad classes may still serve a role as a "fast match" layer in recognition experiments, where their predictions could no longer be certain, due to recognition errors.

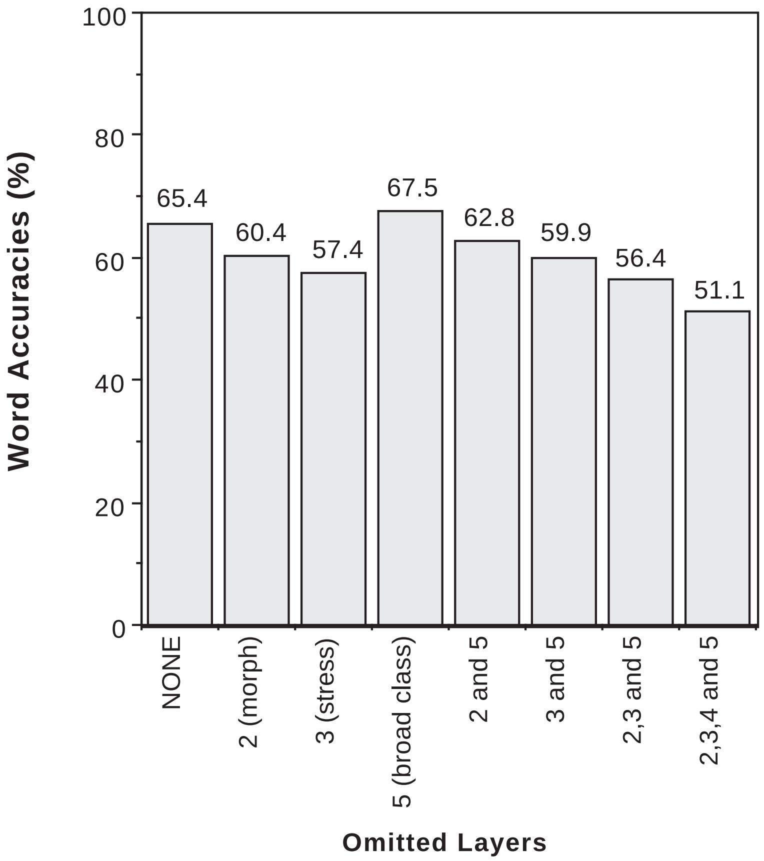


Figure 5-1: Word accuracies as a function of the different layers omitted from the hierarchical lexical representation. Layer 4 is the layer of subsyllabic units.

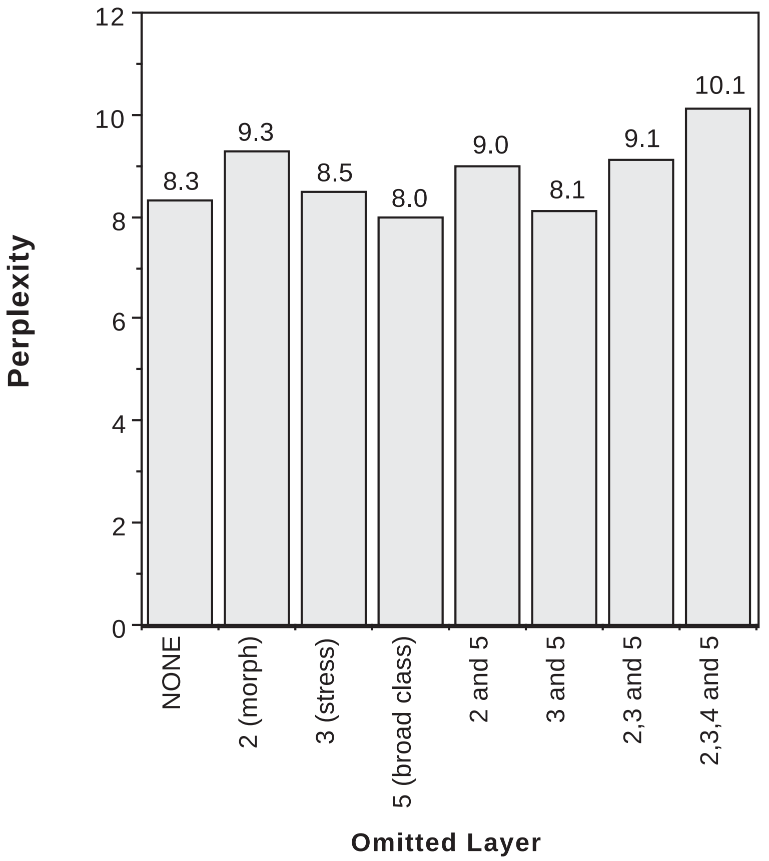


Figure 5-2: Perplexities as a function of the different layers omitted from the hierarchical lexical representation. Layer 4 is the layer of subsyllabic units.

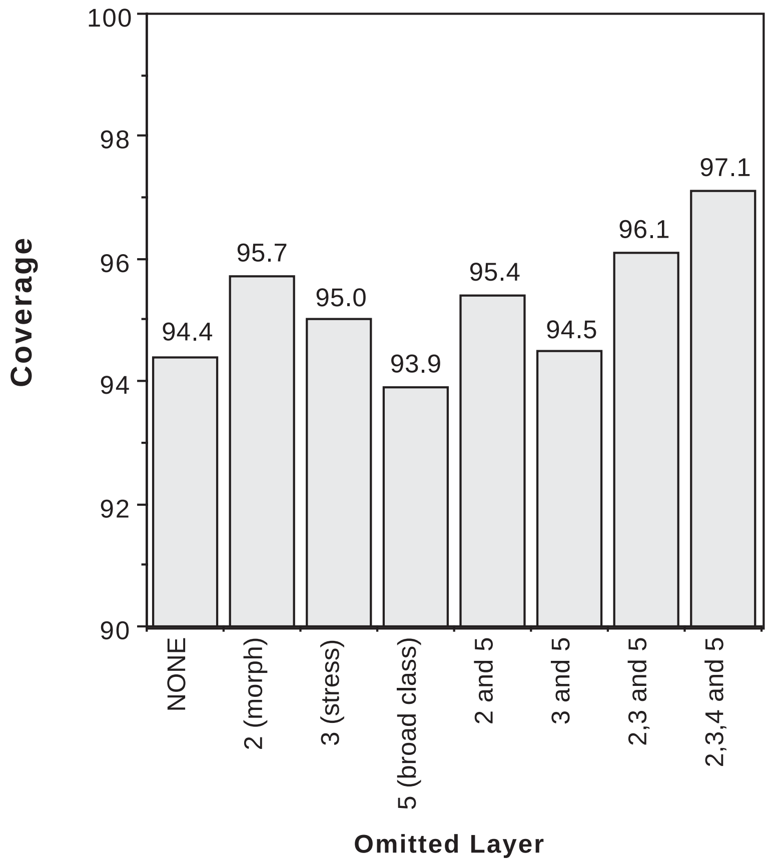


Figure 5-3: Coverage as a function of the different layers omitted from the hierarchical lexical representation. Layer 4 is the layer of subsyllabic units.

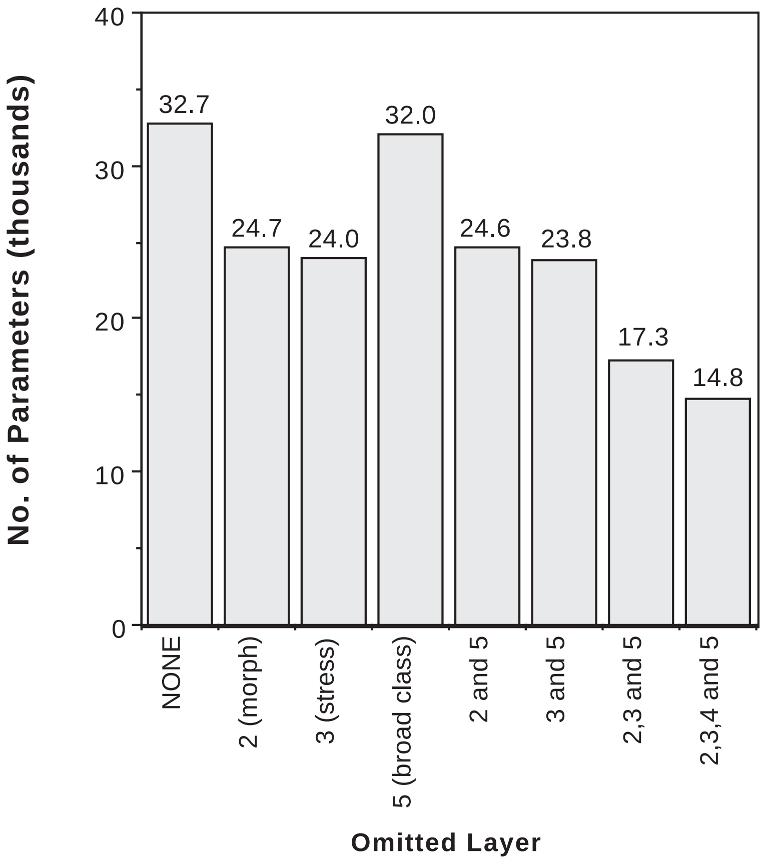


Figure 5-4: Number of parameters as a function of the different layers omitted from the hierarchical lexical representation. Layer 4 is the layer of subsyllabic units.

inclusion of the broad class layer probably led to too much smoothing across the individual phonemes within each broad class, resulting in inferior performances.

Since we have discovered that omitting the layer of broad classes leads to slight performance improvement, this reduced configuration is maintained in all of our subsequent experiments.

5.2 The Non-linguistic Approach

We also compared our current hierarchical framework with an alternative approach which excludes higher level linguistic knowledge. This non-linguistic approach performs transliteration using local letter context, and therefore bears resemblances with the case-base approaches and psychological approaches presented in Chapter 1. The approach is also designed to mirror the layered bigrams with fragment bigram constraints, as will be explained in this section.

The word is represented mainly by a spelling and an aligned phonemic transcription, using the /null/ phoneme for silent letters. Consonant phonemes for geminate letters, however, are duplicated. In general, alignment complies with the training parse trees from the hierarchical approach. For instance, “bright” is transcribed as /b r ɑʸ null null t/, and “spaghetti” as /s p ɨ g null ɛ t t i/.[2] The word is then fragmented exhaustively to obtain letter sequences (word fragments) shorter than a set maximum length. During training, bigram probabilities and phonemic transcription probabilities are then computed for each letter sequence. Therefore this approach captures some graphemic constraints within each word fragment, but higher level linguistic knowledge is not explicitly incorporated. Letter-to-sound generation is accomplished by finding the “best” concatenation of letter sequences which constitutes the spelling of the test word. Mathematically, let l denote the spelling of the test word, an s_i denote a letter sequence (or word fragment) with t_i being its most

[2] The last vowel here is the unstressed /i/.

probable phonemic transcription. Furthermore, let S be a possible concatenation which constitutes l, i.e. $l = S = s_1 s_2 ... s_n$ which corresponds to the phonemic transcription $T = t_1 t_2 ... t_n$. The spelling-to-pronunciation generation process can then be represented as:

$$\begin{aligned} T &= \max_S \; P(T \mid S,\; l)\; P(S \mid l) \\ &= \max_S \prod_{i=1}^{n} P(t_i \mid s_i,\; l)\; P(s_i \mid s_{i-1},\; l)^{\alpha} \end{aligned} \tag{5.2}$$

In the above equation, α is a weighting factor for the language score and its value was optimized using the training data. A bigram language model is used for the letter sequences. In going from the first step to the second in Equation 5.2, we have assumed that the prediction of the next letter sequence is dependent on the current letter sequence only. This is purposely designed to conform with the layered bigrams, where the *current column* is the only context applied in predicting the *next column*. Another assumption is that the phonemic transcription of a letter sequence is independent of the context outside the letter sequence itself, so that each letter sequence is directly mapped to its *single* most probable phonemic transcription. The testing procedure uses a Viterbi search to find the most probable segmentation for the spelling of a word, drawing lexical analogies while capturing context through the use of longer letter sequences, and eventually derives the generated phonemic transcription from the top-scoring segmentation. Some examples of generated outputs from the non-linguistic approach are tabulated in Table 5.1.

To ensure a fair comparison with the hierarchical approach, we use the same training set and development test set to run spelling-to-pronunciation generation experiments with the non-linguistic approach. Duplicated phonemes and the /null/ phoneme are removed from the generated pronunciations before matching against reference pronunciations. Several different value settings were used for the maximum

Word	Segmentation	Generated Pronunciation
acquire	#ACQUIR+E#	/ɪ k k w ɑ^y ɝ/+/null/
bothered	#BOTH+ERED#	/b o θ null/+/ɝ null d null/
bulletin	#BULLE+TIN+#	/b ʊ l l ɪ/+/t ɪ ŋ null/
enjoyment	#ENJOY+MENT#	/ɪ n ǰ ɔ^y null/+/m ɪ n t/

Table 5.1: Examples of generated outputs using the non-linguistic approach

Max. Word Fragment Length	Word Accuracy	Perplexity	No. of Params.
4	60.5%	14.8	303,300
5	67.1%	13.9	508,000
6	69.1%	13.2	693,300

Table 5.2: Experimental results for spelling-to-pronunciation generation using the non-linguistic approach

word fragment length. Generation accuracy is expected to improve as the maximum word fragment length increases, because longer letter sequences can capture more context. This should also be accompanied by an increase in the number of system parameters due to the combinatorics of the letter sequences.

5.2.1 Results

The results for letter-to-sound generation using the non-linguistic approach are shown in Table 5.2. The system includes 26 letters and 58 phonemes. The 6 extra phonemes are: the /null/ phoneme, the pseudo phoneme /ə k/ which can map to the second letter in "mcclellan", as well as 4 pseudo affricates, /k š/, /ŋ z/, /t z/ and /k s/, mapping respectively to the third letter in "luxury", "anxiety", "nazi" and "taxi". Thus, letter-to-phoneme alignment is one-to-one, and no grapheme place-holder is needed. There are no cases of nonparsability in testing because the non-linguistic approach can "back off" to mapping individual letters to their most probable phonemes.

Experiments were conducted with maximum word fragment lengths of 4, 5 and

6 respectively.[3] As a comparison, the two longest graphemes in the hierarchical approach are 4 letters long – "ough" and "eigh", which are pronounced respectively as /o/ and /e/ in "dough" and "eight".

We also conducted a baseline experiment for the non-linguistic approach by setting maximum fragment length to 1 (i.e., the pronunciation of a word is obtained by mapping each letter in the spelling to its most probable phoneme). This gave a word accuracy of 0.8%, which indirectly shows that the non-linguistic approach relies heavily on the use of long word fragments to capture contextual information for letter-to-phoneme mappings.[4]

Table 5.3 shows some erroneous outputs of the single-layered approach. These errors seem to be mostly a result of: (i) forcing a one-to-one letter-to-phoneme mapping, and (ii) the exhaustive fragmentation of the word spelling regardless of the higher level linguistic organization of the word, e.g., its syllabification. For instance, the generated pronunciation for "bubble" lacks a syllable, and ends with the phoneme sequence /b l/ which is not licit in English. The letter sequence "th" at the beginning of a word like "thigh" should be a syllable onset, which is often pronounced as /ð/ or /θ/, but never the phoneme pair /t h/. Another example is the word "suds", in which the letter sequence "uds" is transcribed as /d z/. Here, an analogy is drawn from "clouds" /k l ɑʷ null d z/ but the letters "ou" which should together map to the syllable nucleus have been split. These kinds of errors are not found for the hierarchical approach, because they are precluded by the higher level linguistic constraint in the hierarchical framework.

The highest word accuracy obtained from the hierarchical approach (67.5%)[5] outperforms the single-layer approach with maximum word fragment length set at 4

[3] We did not investigate cases where maximum word fragment lengths are set beyond 6, due to computational limitations, and the vast number of training parameters required.

[4] The mean fragment length used for the test set (with maximum fragment length set at 6) was 3.7, while the mean grapheme length in the hierarchical approach was 1.2.

[5] 6% of the errors in the hierarchical approach are due to parse failure. The next chapter presents a number of robust parsing strategies to overcome this problem.

Word	Segmentation	Generated Pronunciation
bubble	#B+UBB+L+E#	/b/+/ʌ b b/+/l/+/null/
suds	#S+UDS#	/s/+/null d *z*/
thigh	#T+HIGH+#	/t/+/h ɑʸ null null/

Table 5.3: Error examples made by the non-linguistic approach

and 5, but lies below that with maximum length set at 6. However, the hierarchical approach is capable of reversible generation using about 32,000 parameters, while the single-layer approach requires 693,300 parameters for uni-directional spelling-to-pronunciation generation. In order to achieve reversibility, the number of parameters needs to be doubled.

Given an input spelling, the hierarchical approach may generate *multiple* parses with the correct pronunciation. Summing the probabilities of all the relevant parse theories is computationally prohibitive, hence the perplexity values shown in Figure 5-2 are computed from the top-scoring parse theory *only*, and form upper bounds of the *true* perplexity values. The hierarchical approach obtains its lowest perplexity value of 8.0 while omitting the broad class layer, and the single-layer approach obtains 13.2 with the maximum fragment length set at 6. Though the hierarchical approach seems to provide more constraints, direct comparison of these perplexity values may not be entirely fair because the two approaches divide their probability spaces in very different ways. Perhaps a better comparison is to assume up front that partial information (e.g. segmentation) about the top-scoring parse is provided, and focus on the corresponding probability subspace. In this case, the perplexity values computed for the hierarchical approach and the single-layer approach are 5.3 and 7.7 respectively. The hierarchical approach is still substantially more constraining.

5.3 Chapter Summary

The comparative experiments reported in this chapter have shown that each layer in the hierarchy provides additional constraints for generation, which contributes towards higher generation accuracy and lower perplexity. The only exception is the broad class layer. Its inclusion seems to relax constraints and extend coverage by sharing training data to a higher degree. The different layers also interact together to provide a parsimonious description of English orthographic-phonological regularities. We made a comparison, based on spelling-to-pronunciation generation, between the hierarchical lexical representation coupled with an inadmissible stack decoding search, and an alternative, non-linguistic representation conjoined with a Viterbi search. By virtue of the incorporated linguistic knowledge, the former has attained reversibility and comparable performance with 20 times fewer parameters than the latter. In the next chapter, we will attempt to extend the coverage of the parser in order to address the sparse data problem.

Chapter 6

Robust Parsing

In a parsing paradigm there is a constant tradeoff between providing parsing constraints and obtaining sufficient coverage. Constraints are important to prevent overgeneration, but in order to account for previously unseen structures, it is necessary to relax certain constraints and generalize. In this chapter, we will describe our attempt to increase the coverage of our parser, so as to handle the "nonparsable" words mentioned in previous chapters. These words are "nonparsable" because our original parser cannot generate a complete parse tree based on the input spelling / phonemic pronunciation, and hence no output is obtained from the letter-to-sound / sound-to-letter generation task. In letter-to-sound generation, about 6% of the development test set was nonparsable, and in sound-to-letter generation, about 5% was nonparsable.[1] Examples of the nonparsable words can be found in Appendix G. We have augmented our parser with a "robust parsing" capability in order to deal with these problematic words. In the upcoming sections, we will first describe the causes of parse failure, followed by the architecture of our robust parser and finally the performance improvement brought about by robust parsing.

[1] These percentages are based on experiments which have the layer of broad classes omitted from the hierarchy.

6.1 The Causes of Parse Failure

The sparse training data problem aggravates as we descend the hierarchy, because terminal categories and nonterminal categories at the lower levels are more specific. For example, the syllable-part category [ONSET] consists of subcategories including all the consonant phonemes and grapheme/letter terminals that occur in the syllable onset position. Hence, there is a conglomerate of statistics in the training data that relates to the [ONSET] category and therefore the associated probabilities should be quite robustly trained. Comparatively speaking, a grapheme terminal such as #[que] in "critique" has a higher degree of specificity and occurs less frequently. Other specific categories may not appear in the training data at all.

Analysis of the nonparsable words in both letter-to-sound and sound-to-letter generation has shown that the main cause of parse failure is zero advancement probabilities, where the current column history cannot advance to the next right terminal, i.e.

$$P(\mathit{next\ terminal} \mid \mathit{current\ history}) = 0 \tag{6.1}$$

This is true for generation in *either* direction, and leads to much overlap between the two sets of nonparsable words (about one-third of the words in either set). Based on our analysis, we have characterized three conditions under which zero advancement probabilities occur:

1. Compound words — words such as "cocktail," "everyday" and "typewriter" etc., contain a "word boundary." Since our training set consists mainly of simple words, the advancement from the column to the left of the word boundary to the next grapheme terminal to the right of the word boundary often has zero probability. An example can be found in the word "typewriter," for which we would expect the output parse tree to be as shown in Figure 6-1. However, in

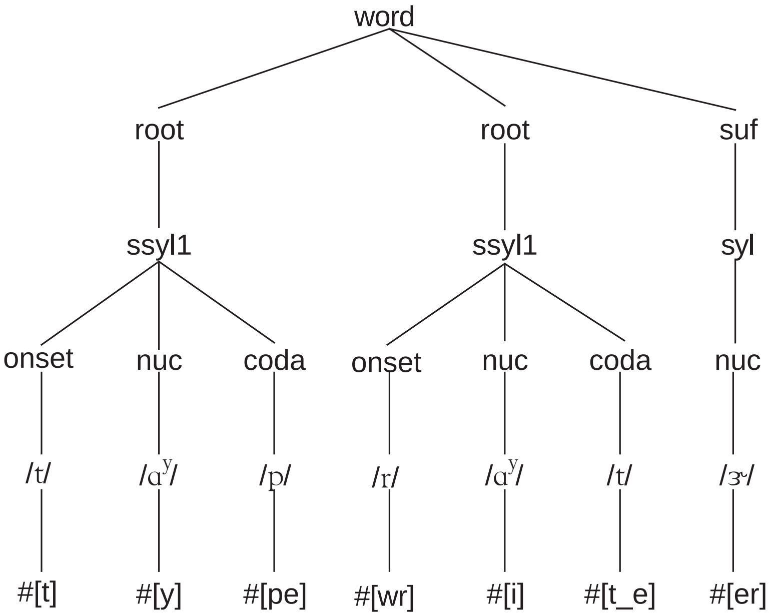


Figure 6-1: Parse tree for the word "typewriter."

letter-to-sound generation, the advancement from the column history {WORD ROOT SSYL1 CODA /p/ #[pe]} to the next terminal #[wr] has no previous instances in the training data. With direct symmetry, parse failure also occurred for the same reason in sound-to-letter generation, where the input pronunciation is /t ɑʸ p r ɑʸ t ɝ/. Amongst the grapheme terminals which correspond to the phoneme /p/, none can advance from left to right to another grapheme terminal which predicts the phoneme /r/ bottom-up.

2. New grapheme terminals — we sometimes encounter new grapheme terminals in the test set and these terminals are not associated with any trained probabilities. For example, in the word "sioux," pronounced as /s u/, the last four letters in the spelling should correspond to the second phoneme in the pronunciation. The parser begins with the terminal #[s], mapping it to /s/), with several distinct upper histories. Since the training data does not contain the terminal #[ioux], the parser advances to the second terminal #[i], mapping it to /ɑʸ/, /y/ and /ɪ/.

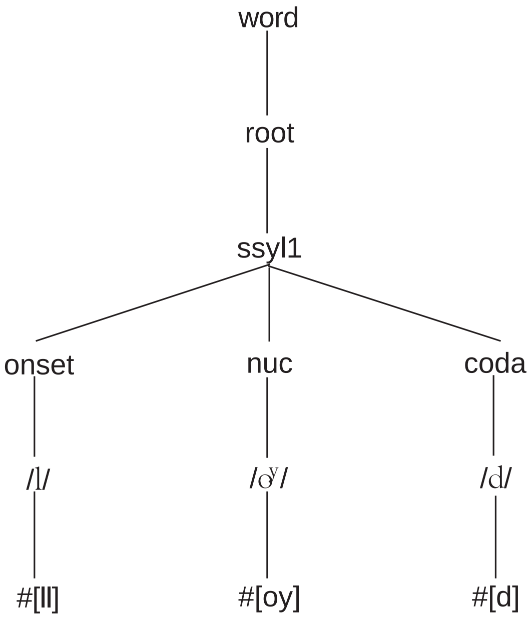


Figure 6-2: Parse tree for the word “lloyd.”

Amongst these, only the columns containing {/ɑʸ/ #[i]} could advance further to the third terminal #[o], which is pronounced as either /ɪ/ or the unstressed /o/. However, none of these columns can push through to the next terminal #[u], which led to a parse failure. Another example of new grapheme terminals is the case of geminate letters, such as in the word “lloyd.” The parse tree for “lloyd” should be as shown in Figure 6-2. The parse begins with the grapheme #[ll], and the geminate letters correspond to a single phoneme /l/. However, the grapheme #[ll] has not been observed to start a word or advance to #[oy] in the training data, and this has rendered the word “lloyd” nonparsable. Similarly, parse failure in “embassy,” “luggage,” and “settling” are due to geminate letters. The pronunciations of these words, however, are parsable in sound-to-letter generation. The output spellings do not contain geminate letters. For example, “loid” was generated from /l ɔʸ d/ (pronunciation for “lloyd”), “embicy” was generated from /ɛ m b ɪ s i/ (pronunciation for “embassy”), “lugage” was

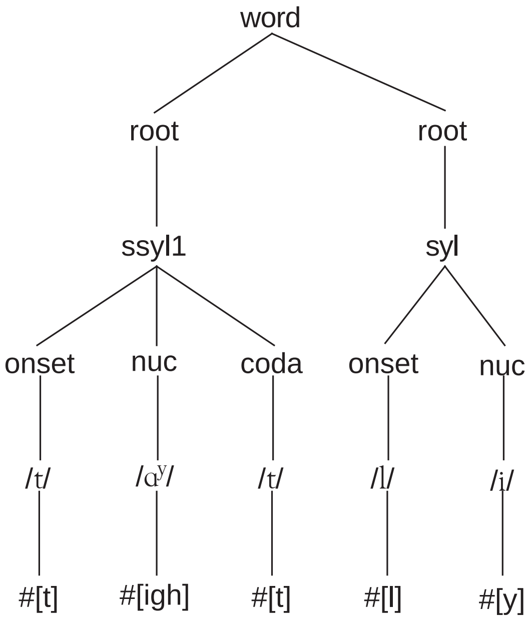


Figure 6-3: Parse tree for the word "tightly."

generated from /l ʌ g ɪ ǰ/ (pronunciation for "luggage"), and "setling" was generated from /s ɛ t l ɪ ŋ/ (pronunciation for "settling").

3. Sparse training data — the third condition has to do with sparse training data problems which do not befall the previous two conditions. For example, "tightly" showed up as a nonparsable word in letter-to-sound generation. The parse tree for "tightly" should be as shown in Figure 6-3. Parse failure is caused by the zero advancement probability from the column history {WORD ROOT SSYL1 ONSET /t/ #[t]} to the next grapheme terminal #[igh]. In sound-to-letter generation, the spelling outputs for the phoneme sequence /t ɑʸ t l i/ include "titely," "tytely" and "teitly." Another illustrative example is provided by the word "cushion." In letter-to-sound generation, the output pronunciation was /k ʌ š ɪ n/. In sound-to-letter generation with the input pronunciation /k ʊ š ɪ n/, the first column with history {WORD ROOT SSYL ONSET /k/ #[c]} needs to advance to a grapheme terminal that can correspond to the second

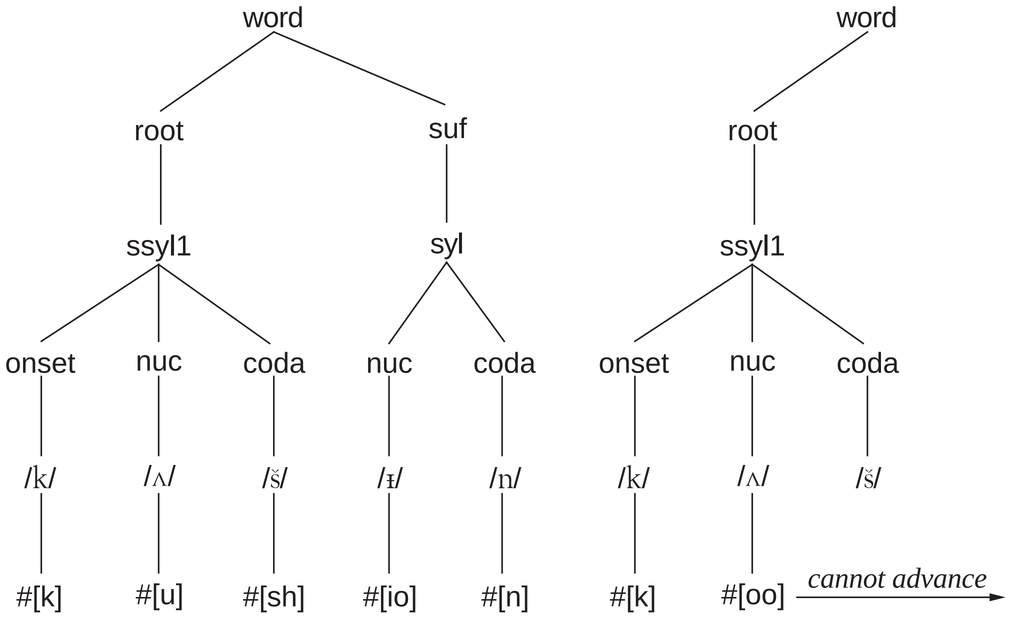


Figure 6-4: Parse trees for the word "cushion" — (left) from letter-to-sound generation and (right) from sound-to-letter generation.

phoneme /ʊ/. According to the trained probabilities, the only such grapheme terminal is #[oo]. However, a column with history {WORD ROOT SSYL NUCLEUS /ʊ/ #[oo]} cannot advance to a grapheme terminal corresponding to the third phoneme /š/. Hence a parse failure results. This is illustrated in Figure 6-4.

Our robust parsing strategy is designed specifically for dealing with these three conditions. A detailed description of the robust parser is provided in the next section.

6.2 The Robust Parser

The top-level architecture of our robust parser is shown in Figure 6-5. It contains the basic modules of bottom-up prediction, left-to-right advancement and a stack, which is similar to the original parser. We have augmented the regular path in left-to-right advancement with three other options, labelled as "end-start," "skip," and "partial-history," in order to handle the three causes of parse failure as characterized from the nonparsable words in the previous experiments. Typically, when a column advances to the next terminal, each of the four options can be selected with finite probability. The values of these probabilities are empirically set to sum to 1, and to favor the regular advancement path. The respective values are: $P(\text{regular}) = 0.94$, $P(\text{end-start}) = 0.025$, $P(\text{skip}) = 0.025$, $P(\text{partial-history}) = 0.01$. The mechanisms which take place along each of the three additional advancement paths are described as follows:

1. End-Start — this advancement path expedites analysis of compound words. It allows the parser to end a parse tree in the middle of a word spelling or phonemic transcription and start a new parse for the remaining part of the input. This is illustrated by the parse tree in Figure 6-6, which is the robust parse output for the word "typewriter." In extending the left-history {WORD ROOT SSYL1 CODA /p/ #[pe]} to the "robust-end" node, the advancement probability is computed as the product (or the sum of the logarithms) of the probability of extending to [END] and the "end-start" penalty, i.e.

$$\begin{aligned} P_{robust\ end\ advancement} &= P(\mathit{next\ terminal} = [\textsc{end}] \mid \ \mathit{left\text{-}history}) \\ &\quad * P(\mathit{end\text{-}start}) \end{aligned}$$

   Subsequently, in extending the robust-end node to the next grapheme terminal #[wr], the advancement probability is simply the unigram probability of starting a parse with #[wr], i.e.

$$P_{robust\ start\ advancement} = P_{start\ unigram}(\#[\text{wr}])$$

   The new terminal node (with an empty history) created for #[wr] is then pushed onto the stack.

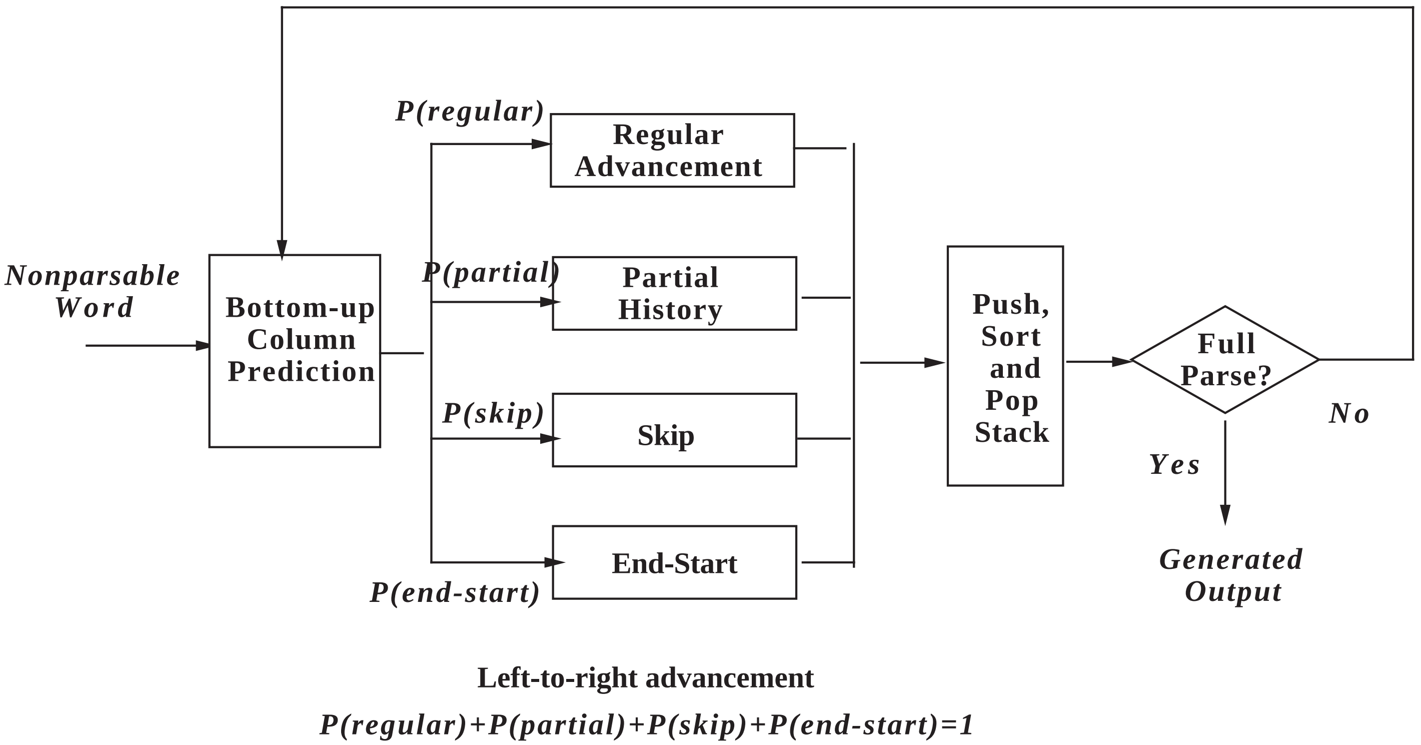


Figure 6-5: Top-level architecture for the robust parser.

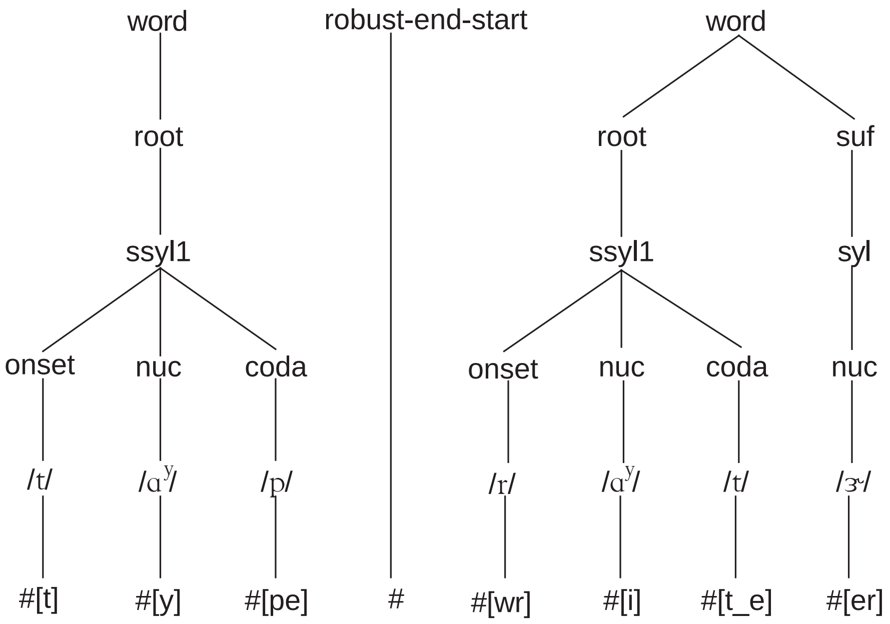


Figure 6-6: Robust parser output for the word "typewriter."

2. Skip — this advancement path deals with grapheme terminals which consist of two geminate letters, e.g. #[gg] in "luggage" and #[ss] in "embassy." Consonant phonemes can be mapped to their corresponding letter terminals or geminate letter terminals.[2] Consequently, the robust parser is designed to allow skipping of one of the two geminate letters, and the word is parsed as though the gemination is replaced by a single letter. As an illustration, Figure 6-7 shows the output parse tree from the robust parser for the word "lloyd." In advancing the column history {WORD ROOT SSYL1 ONSET /l/ #[l]} to the next terminal "#[l]-skip," the probability involved is simply the "skip" penalty $P(skip)$, i.e.,

$$P_{robust\ skip\ advancement} \;=\; P(skip)$$

and the new terminal node for "#[l]-skip" is pushed onto the stack.

[2] Note that a single letter grapheme which corresponds to a vowel is generally pronounced differently from its geminate counterpart, e.g. #[o] is pronounced differently from #[oo], but #[s] and #[ss] are pronounced identically.

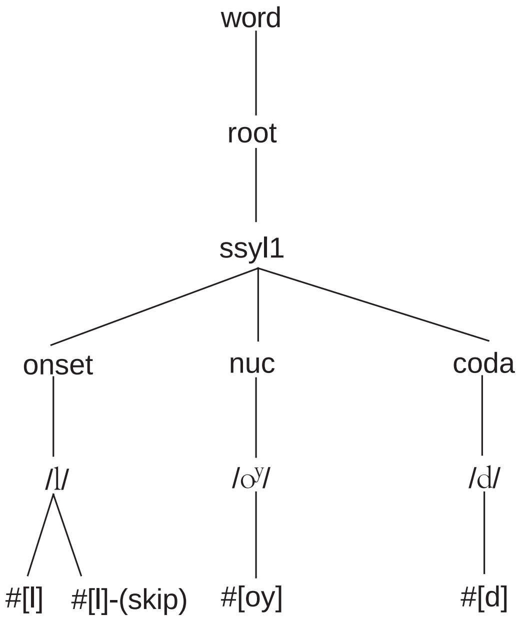


Figure 6-7: Robust parser output for the word "lloyd."

3. Partial — this path slackens constraints when the parser encounters zero advancement probabilities outside the two previously mentioned conditions. Consider the parsable word "lightly," whose parse tree is shown in Figure 6-8. The history in the first column, {WORD ROOT SSYL1 ONSET /l/ #[l]} can transition to the next grapheme #[igh] with non-zero probability. However, if we substitute the phoneme and grapheme in the history to be /t/ #[t] respectively, as in "tightly," then the transition probability to #[igh] becomes zero, resulting in parse failure. In the robust parser, we circumvent such sparse data problem by sharing advancement probabilities across phonemes and graphemes which belong to the same syllable part category. This is accomplished by "backing-off" to the syllable part level. As a result, the probabilities are conditioned only on the *partial* left-history, i.e.

$$P(\#[\text{igh}] \mid \{\textsc{word root ssyl1 onset } /\text{t}/\ \#[\text{t}]\} = 0$$

but

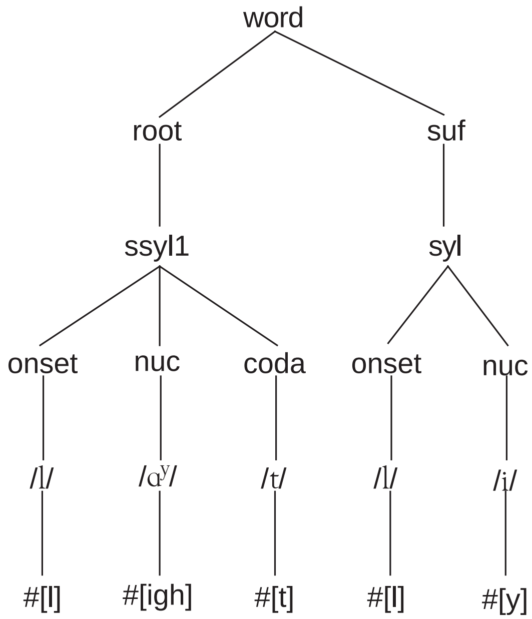


Figure 6-8: Parse tree for the word "lightly."

$$P(\#[\text{igh}] \mid \{\text{WORD ROOT SSYL1 ONSET}\}) \; > \; 0$$

The advancement from {WORD ROOT SSYL1 ONSET /t/ #[t]} to #[igh] incurs both the "backoff" advancement probability as well as the "backoff" penalty, i.e.,

$$\begin{aligned} P_{partial\ history\ advancement} \;\; = \;\; & P(\#[\text{igh}] \mid \{\text{WORD ROOT SSYL1 ONSET}\}) \\ & * \, P(partial) \end{aligned}$$

The new node created for #[igh] is then pushed onto the stack.

During robust parsing, the parser attempts to advance partial theories along all four paths (regular, end-start, skip and partial). However, if a given terminal can be reached via a regular advancement path, the partial advancement paths will not be pursued. Sound-to-letter generation may tolerate additional constraint relaxation, which allows the stressed and reduced versions of the vowels /u/, /o/ and /i/ be interchanged.

Task	Coverage	Word Accuracy	Phoneme/Letter Accuracy
Letter-to-sound (original)	94%	67.5%	87.6%
Letter-to-sound (robust)	100%	69.2%	91.3%
Sound-to-letter (original)	95%	52.9%	83.8%
Sound-to-letter (robust)	100%	53.7%	88.5%

Table 6.1: Performance improvement on the development test set with the addition of robust parsing. Zero accuracies were given to nonparsable words.

6.3 Performance

The additional advancement options in the robust parser enable it to propose and explore many more theories at a given instant than the original parser. Therefore, due to efficiency concerns, we revert to the robust parser only when the original parser fails. We have also increased the limit on the number of stack-pops from 330 to 3000 to attune to the needs of robust parsing. This new coupled configuration has attained complete coverage of the development test set, and brought about a slight performance improvement in the development test set as tabulated in Table 6.1. The nonparsable words in the previous experiments (which did not attain complete coverage) had been given a word accuracy and phoneme/letter accuracy of zero.[3]

The performance on the real test set is tabulated in Table 6.2. Using only the original parser, about 7% of the real test set was nonparsable in letter-to-sound generation, and the corresponding value for sound-to-letter generation was 6%. Complete coverage was attained with the inclusion of the robust parser.

In general, the word accuracies on the development test set are about 2-3% higher than those of the real test set, and the phoneme/letter accuracies are about 1% higher. Similar trends are observed in *both* test sets when robust parsing is incorporated.

[3] The percentage of nonparsable words in the development test set for sound-to-letter generation rose from 4% to 5% upon omitting the layer of broad classes, while that for letter-to-sound generation remains the same.

Task	Coverage	Word Accuracy	Phoneme/Letter Accuracy
Letter-to-sound (original)	93%	65.0%	86.9%
Letter-to-sound (robust)	100%	66.3%	90.5%
Sound-to-letter (original)	94%	51.0%	82.6%
Sound-to-letter (robust)	100%	52.0%	87.9%

Table 6.2: Performance improvement on the development test set with the addition of robust parsing. Zero accuracies were given to nonparsable words.

Robust parsing has brought about slight improvements in word accuracies (about 1%), phoneme accuracies (about 3%) and letter accuracies (about 5%). The increment is small because the nonparsable words seem to be a difficult subset. Analysis of the nonparsable words in the development test set shows that the robust parser achieved a 28% word accuracy and 69.5% phoneme accuracy in letter-to-sound generation. Figures 6-9 to 6-11 show examples of some errors in robust parsing. A "robust word boundary" was inserted wrongly in "charlie," "henrietta" and "joe." In particular, the correct parse for "joe," which gave the pronunciation /ǰ o/, was ranked fourth on the stack.

Similar analysis of the nonparsable words in the development test set from sound-to-letter-generation showed that robust parsing gave a 15.6% word accuracy and 75.5% letter accuracy. We have also included some examples of the robust parsing errors in Figures 6-12 to 6-14. A comparison of Figures 6-4 and 6-12 shows that with robust parsing, the grapheme terminal #[oo] can now advance to #[sh] which is under the phoneme /š/. The generated parse trees for "henrietta" and "typewriter" in sound-to-letter generation are different from those in letter-to-sound generation. A robust word boundary is wrongly inserted in the two parse trees shown in Figures 6-13 and 6-14.

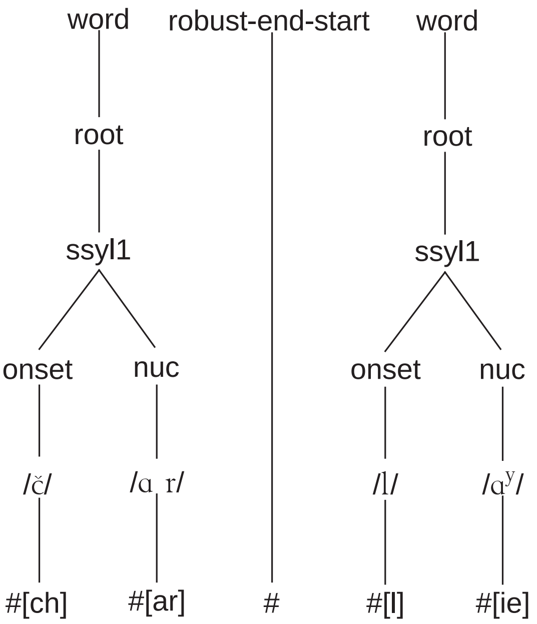


Figure 6-9: Parse tree for the word “charlie” from robust letter-to-sound generation.

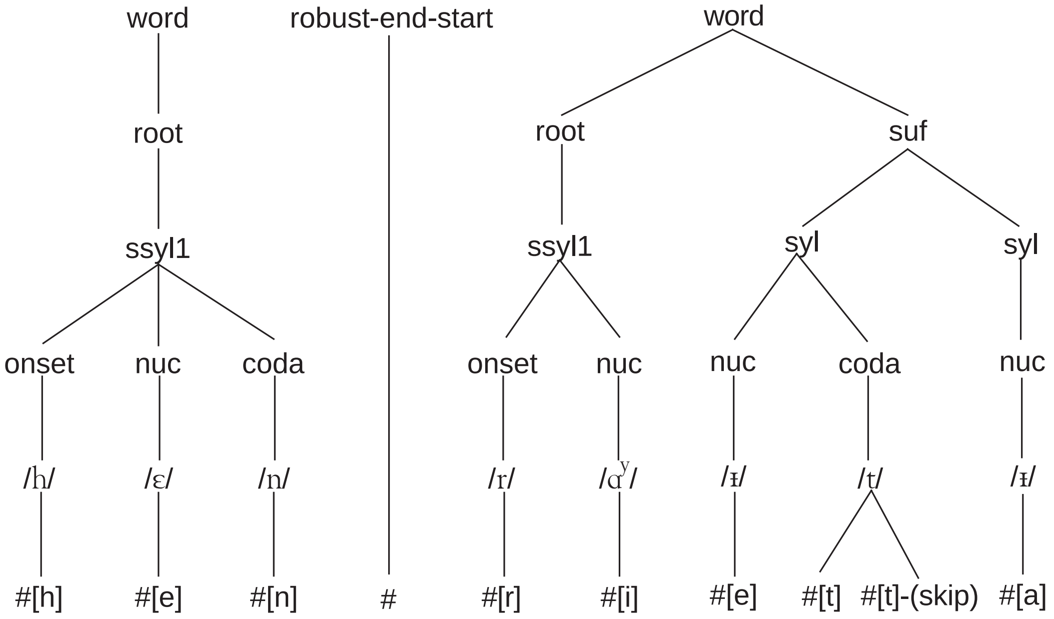


Figure 6-10: Parse tree for the word “henrietta” from robust letter-to-sound generation.

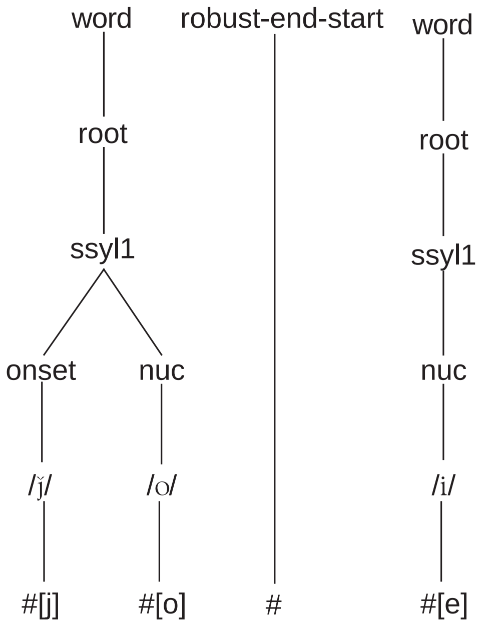


Figure 6-11: Parse tree for the word "joe" from robust letter-to-sound generation.

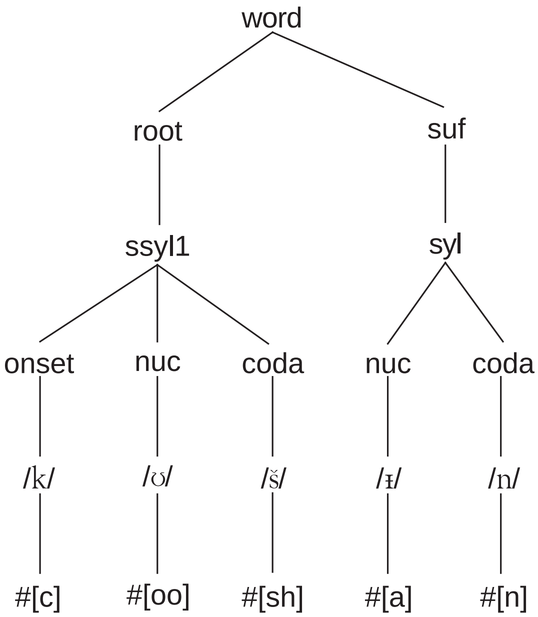


Figure 6-12: Parse tree for the word "cushion" from robust sound-to-letter generation.

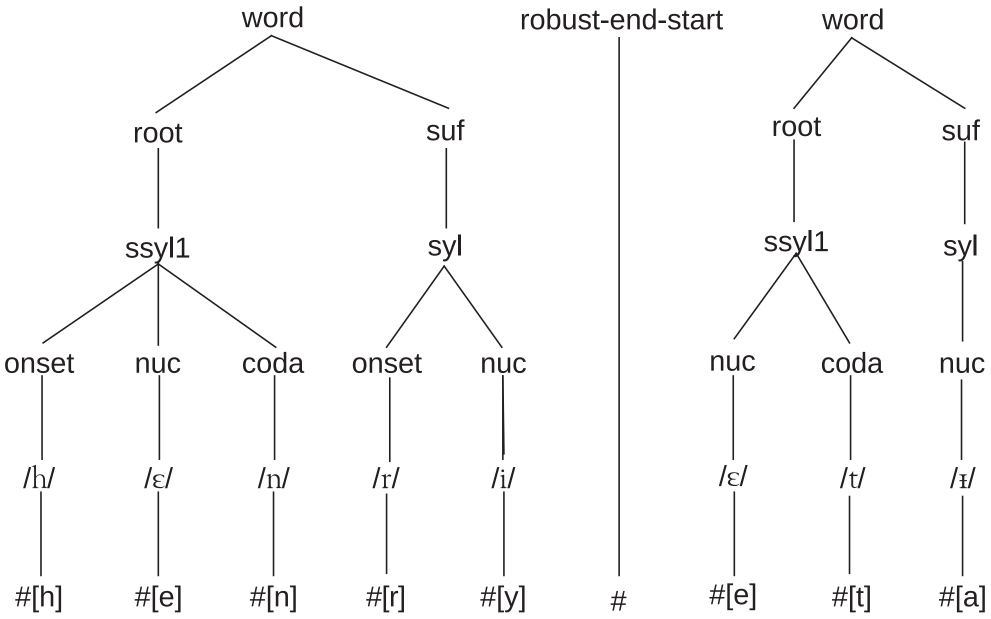


Figure 6-13: Parse tree for the word "henrietta" from robust sound-to-letter generation.

6.4 Chapter Summary

In this chapter, we have described the design of a robust parser, which is used in association with the original parser to enlarge the overall coverage of test data. The three main robust parsing mechanisms are: (i) inserting a word boundary in the parse to handle compound words, (ii) skipping one letter in a geminate pair to deal with new grapheme terminals of geminates and (iii) conditioning advancement probabilities upon the *partial* left-history instead of the *entire* left-history when zero advancement probabilities are encountered. These extensions have brought about complete coverage as well as a slight performance improvement in both the development test set and the real test data.

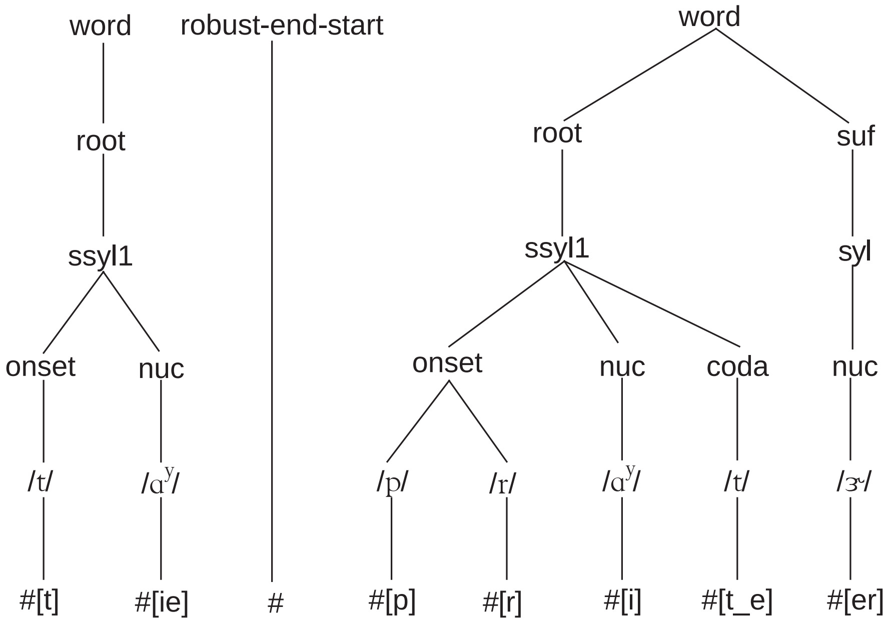


Figure 6-14: Parse tree for the word "typewriter" from robust sound-to-letter generation.

Chapter 7

Extending the Hierarchy

We have thus far focused on a hierarchical structure for spoken English which consists of multiple levels of linguistic representation, ranging from the generic English WORD, through the intermediate categories of morphology and syllables, to the finer categories of phonemes and graphemes. However, as mentioned in the beginning of the thesis, this hierarchical representation could be extended to encompass natural language constraints, prosodic information and dialog modelling constraints on top, as well as phonetics and acoustics below. In this chapter, we have taken a first step towards demonstrating that the hierarchy is extendable. We have added a layer of phones beneath the layer of phonemes in the hierarchy. Consequently, the terminal layer in our hierarchical representation becomes dual in nature[1] — it can be a layer of phones *or* letters. Using the augmented hierarchical representation with the layered bigram probabilities, we are able to capture some dialectal and phonological variations both *within* a word and *across* word boundaries.

[1] The phonetic and graphemic layers both reside beneath the phoneme layer.

7.1 Background

In the past, handling phonological variations in the development of speech recognition sytems has been accomplished mainly by phoneticians with a set of rewrite rules [62] [99] that explains context dependencies. These rules transform phonemes into phones, and phones into other phones. The identity of the phoneme/phone sequence prior to a transformation is often not preserved, rendering the transformation irreversible. These transformations create alternate word pronunciations in the process. The pronunciations are then matched against a speech recognizer output. A large number of rules is often required to capture allophonic variations, as well as within- and across-word phonological variations. Furthermore, the *ordering* of the rules in the set is necessary and important for arriving at the correct phonetic output. This approach is hampered by the difficulty in maintaining the ordered rules, especially when the situation calls for the addition of a new rule. Determining the position of a new rule requires close examination of how the rules interact with one another. This involves following through successive rule applications, which is a difficult task because the rule transformations are irreversible and this makes back-tracing either tedious or impossible.

A previous attempt has also been made which attaches probabilities to the rules productions [93]. The probability of a given rule reflects the number of times the rule is actually applied, normalized over the number of times the rule can be applied. Each rule is compiled to form a series of rule clauses specifying the rule's transformations on a phone lattice under different contexts. Upon rule application, the rule probability will be incorporated into the arc probabilities of the phone lattice. However, these rule probabilities assume that rule applications are independent of one another, which is not valid for *ordered* rules. Instead, these production probabilities should be dependent on the preceding rule productions, which may perhaps make the probabilities more complicated and less trainable.

7.2 Motivation

We feel that the layered bigrams procedure will be a useful paradigm for characterizing phonological variation. Much of the information provided by the hierarchical lexical representation can be used to account for phonological variations. For example, letter-to-sound mappings tend to be less predictable at morphological boundaries — the phoneme /p/ in "display" is aspirated, while that in "displace" is not. The identity of the phonemes before transformation is preserved by the layered bigrams framework in the context upon which the probabilities are conditioned. Therefore, phoneme-to-phone transformations that have taken place within a particular context can be clearly observed from the parse tree. The hand-crafted, ordered rewrite rules are replaced by probabilities which can be automatically trained. Hence, the system is unburdened of the tedium of updating and maintaining the ordered rewrite rule set. This has motivated us to extend the hierarchical lexical representation with a phone level beneath the phoneme level. The objective is to illustrate how layered bigrams can potentially be used to capture phonological rules, with the tacit assumption that these phonological rules are conducive to speech recognition.

7.3 Experimental Corpus

To conduct this pilot study, we wanted to concentrate our efforts on a small set of data carefully selected for potential validation of our approach. Rather than complicating the experiment with a recognition task, we have selected as our experimental corpus the "sa" sentences of TIMIT, for which carefully transcribed phonetic labels are available. These sentences are designed especially for the study of dialectal variations. They also provide phoneme-to-phone alignments for generating our training parse trees, and the phonetic transcriptions are labelled manually through listening tests along with visual aids of the spectrogram and waveform [44]. The two "sa" sentences are:

1. “She had your dark suit in greasy wash water all year.”

2. “Don’t ask me to carry an oily rag like that.”

The speakers come from 8 different dialectal regions,[2] namely, New England, Northern, North Midland, South Midland, Southern, New York City, Western and “Army Brat.” In our study, we use 362 training speakers, and a disjoint set of the 24 NIST designated “core” test speakers. The test set consists of 3 speakers from each of the 8 dialectal regions.

7.4 Phonological Variations

The two “sa” sentences are especially designed for the study of phonological variations. In this section, we provide a brief description of such variations as observed in our training data.

Some of the phonological variations in the “sa-1” sentences are illustrated in Figure 7-1. For example, on either side of the word boundary between “had” and “your,” the phoneme /d/ transitioning to the phoneme /y/ can be realized as [č]-[null], [d□]-[y], [d]-[y], [ʕ]-[y], [ǰ]-[null], [ǰ]-[y] or [null]-[y].[3] Therefore, we observe cases where the *alveolar* stop /d/ is *palatalized* to become either the affricate [č] or [ǰ], followed by the semi-vowel /y/, which may be deleted. Another example concerns the vowel in “wash.” It can be realized as [ɔ], [ʊ], [ʌ], [ɝ], or [ɑ]-[r], with retroflexion inserted in the last two cases.

Similarly, we find much variation in the “sa-2” sentences, as illustrated in Figure 7-2. On either side of the word boundary between “Don’t” and “ask,” the phoneme /t/ transitioning to /æ/ can be realized as [t]-[æ], [t□]-[æ], [ʔ]-[æ], [null]-[ɑ], [null]-[ɛ], [null]-[ɑʷ], [null]-[æ], [null]-[e], and [d]-[æ]. Here we observe several different allophones

[2] This is the geographical dialect area where the subject spent the most years between ages 0 and 10.

[3] [null] denotes the NULL phone.

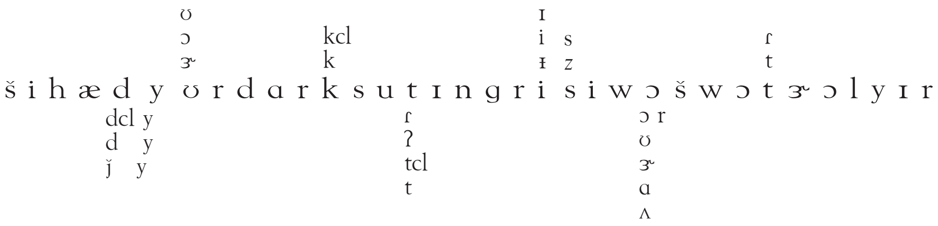

š i h æ d y ʊ r d ɑ r k s u t ɪ n g r i s i w ɔ š w ɔ t ɝ ɔ l y ɪ r
ʊ
ɔ
ɝ
kcl
k
ɪ
i
ɨ
s
z
ɾ
t
dcl y
d y
ǰ y
ɾ
ʔ
tcl
t
ɔ r
ʊ
ɝ
ɑ
ʌ

Figure 7-1: Some phonological variations occurring in the sa-1 training sentences — "She had your dark suit in greasy wash water all year." *dcl* and *kcl* denote d-closure and k-closure respectively.

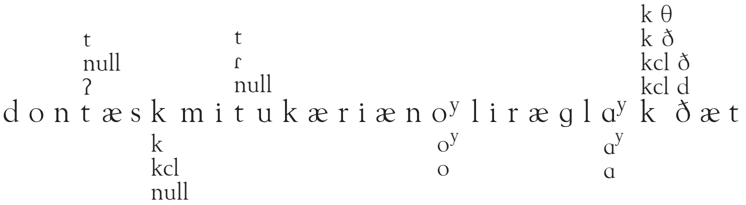


Figure 7-2: Some phonological variations occurring in the sa-2 training sentences — "Don't ask me to carry an oily rag like that." *tcl* and *kcl* denote t-closure and k-closure respectively.

for the phoneme /t/ in "don't." It may be *released* in the phone [t] or [d], *unreleased* in [t□], *glottalized* to become [ʔ], or deleted as in [null]. The subsequent vowel /æ/ may also take many different forms. Finally, the vowel in "like," /ɑʸ/, may assume various identities, including [æ], [ɑʸ], [i], [ɑ] and [ʌ].

We would like to capture such dialectal and phonological variations in the "sa" sentences, by extending *both* the hierarchical lexical representation and the layered bigrams framework. Our experiments attempt to parse the phonetic transcription (with no indication of the word boundary locations) of an "sa" sentence. The purpose is to obtain the sequence of words in the sentence without prior knowledge of the lexicon. There are two artificial aspects in this experimental setup:

1. The experimental data are sparse because only the two "sa" sentences are used.

More data will be needed to validate the applicability of the approach for speech recognition.

2. Only a single phonetic sequence is used as an input. In a real recognition task, the recognizer provides alternative phones for a given segment of speech. Consequently the parser should be less prone to failure.

The objective of these experiments is to project the idea of using the layered bigrams for capturing phonological rules, and as a framework for speech recognition. The proposed experiment is a modest first step, which should be followed by further, more in-depth studies.

7.5 Extending the Hierarchical Representation

The two "sa" sentences together consist of 21 distinct words which constitute the vocabulary of our current task. For each word in our limited vocabulary, we generate a parse tree from the word level down to the phoneme level, using our letter-to-sound generation system. The parse tree is then extended to the phone level based on the phoneme-to-phone alignments provided by TIMIT. The parse tree for an *entire* training sentence is then obtained by concatenating the parse trees for each *word* in the sentence. An example of the training parse tree of an "sa-2" sentence is shown in Figures 7-3 and 7-4.

The extended parse tree has the generic category SENTENCE at the top level, followed by a layer of WORD categories separated by the word boundary terminal #. Some of the phonological variations described in the previous section can be observed from this sentence parse tree. For example, the phoneme /t/ in the word "don't," and the phoneme /k/ in the word "ask" have been deleted, and the vowel /u/ in the word "to" has been devoiced. In the word "carry," the phoneme /k/ illustrates the case where one phoneme is mapped to multiple phones ([k$^{\square}$][4] and [k]), and the phone [ɝ]

[4] k-closures may be denoted as [k$^{\square}$] or [kcl].

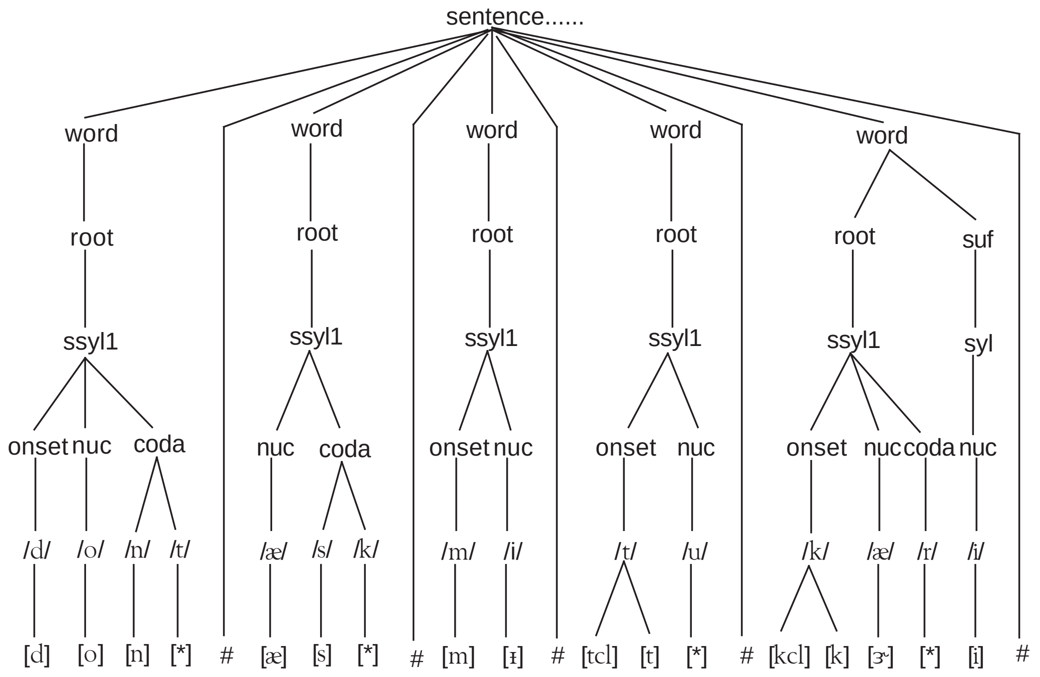


Figure 7-3: The sentence parse tree of the first half of a training "sa-2" sentence — "Don't ask me to carry..." — with a terminal phonetic layer.

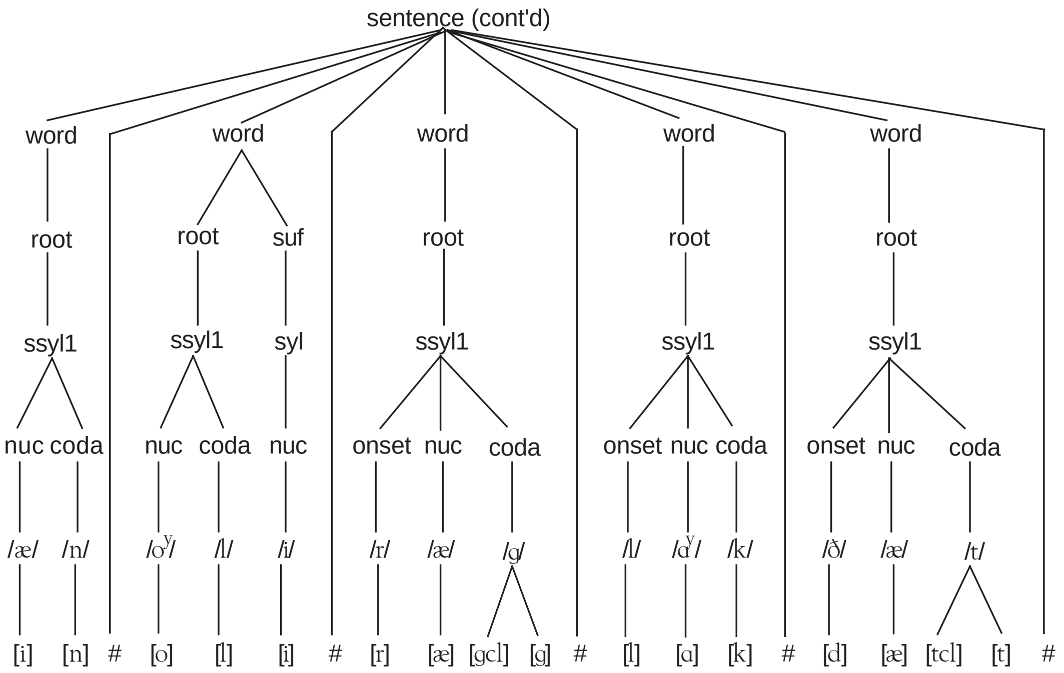


Figure 7-4: The sentence parse tree of the second half of a training "sa-2" sentence — "...an oily rag like that." — with a terminal phonetic layer.

illustrates the case where one phone corresponds to multiple phonemes. Phonological rules specify variations in the arena of the two bottom layers of phonemes and phones, and are extracted automatically and implicitly during training by the layered bigram probabilities.

7.6 Extending the Layered Bigrams Parser

The layered bigrams are extended to parse the phonetic transcription of an *entire* sentence into a sentence parse tree depicted in Figures 7-3 and 7-4. The main augmentations encompass training, robust parsing during testing, as well as lexical access.

7.6.1 Training in the Extended Layered Bigrams

The layered bigram probabilities trained from the training parse trees remain essentially the same, with the addition of the following:

1. Word boundary prediction probabilities

   We have created a special terminal # to denote a word boundary. Similar to other terminal categories, the prediction probability for # is conditioned on the left-history only, i.e.,

   $$P_{word\ boundary\ prediction} = P(next\ terminal\ = \#\mid current\ history) \qquad (7.1)$$

2. Across-word prediction probabilities

   The prediction probability for the phoneme-phone pair to the *right* of a word boundary is conditioned upon the phoneme-phone pair to the *left*. We reckon that the identities of the left-phoneme and left-phone are critical factors in determining the right-phoneme and right-phone. Moreover, the assumption justifies sharing data amongst left phoneme and phone pairs with different histories, which should help alleviate the sparse data problem prevailing at word boundaries.

$$P_{across-word\ prediction} = P(R\ phoneme - phone \mid L\ phoneme - phone) \quad (7.2)$$

7.6.2 Testing in the Extended Layered Bigrams

Testing in the extended layered bigrams amounts to creating a complete parse tree for the phonetic transcription of a testing “sa” sentence. Each phonetic transcription of TIMIT is a single phone sequence, unlike the output of a speech recognizer which provides a phone lattice with alternative phone choices for each time segment. Phone sequences which have not been previously observed in the training data lead to parse failures, unless alternate phone choices are provided. We may therefore be penalizing ourselves by using the phonetic transcription as our input instead of the phone lattice from a speech recognizer. The process of parsing the phones of a sentence is essentially identical to that of parsing the phonemes of a word. The parser operates in a bottom-up, left-to-right fashion, extending each partial parse tree by one column at a time. Word boundaries are predicted during the process. Furthermore, the parser has no knowledge of the vocabulary — only the left-column is used as context for left-to-right advancement, including the advancement to a word boundary terminal #. This is the way we envision the layered bigrams operating with a recognizer, proposing plausible words bottom-up for later verification.

In the event of a partial theory not being able to advance to the next input phone (i.e. the left-to-right advancement probability equals zero), the robust parsing capability allows skipping the phone, and ending the partial parse with a “robust-end” node which incurs a penalty score. The “robust-end” node is subsequently pushed onto the stack. When this node is later popped off the stack for extension, the robust parser proceeds by creating a “robust-start column” based on the input phone following the skipped phone. Bottom-up prediction in the “robust-start column” involves multiplying the *context-independent* probabilities, i.e.,

$$P_{robust\ start\ column} = \prod_{r=7}^{r=2} P(RP_{(r-1)} \mid RS_r,\ LH = \textsc{start}) \tag{7.3}$$

where r is the row-index, $r = 7, 6, 5, ...1$,
RS_r is the right-sibling at row r,
($RS_1 = \textsc{word}$, RS_7=terminal layer)
RP_r is the right-parent at row r.

Using the robust-start column, the parser proceeds to generate a parse tree to account for the rest of the sentence. Figure 7-5 shows an example of a sentence parse tree which contains skipped phones. The parse shows the first half of an "sa-1" sentence, whose phonetic transcription is missing a vowel for "She," and missing retroflexion in the word "your." At both locations, we see a skipped phone, followed by a "robust-end" node, and the parse is continued with a "robust-start" node. Figure 7-6 shows the rest of the sentence.

Once again, the best-first search procedure is used. The first complete parse tree popped off the stack is taken as the output. The sentence parse hypotheses are often subjected to penalty scores due to skipped phones and robust-end nodes. The penalty score is set to a low probability, so the robust partial parse theories will get lower scores and rank low on the stack. In this way the robust parse theories are disfavored and the search mechanism encourages the parser to pursue other theories prior to robust parsing. At the moment, it is unclear how the computation of the actual parse scores for the robust parse theories should include the robust penalties, to contend a fair comparison with other parse theories for N-best rescoring. Consequently, no N-best rescoring based on the actual parse score is used in this study. When this framework is used in recognition experiments, the scoring/search issue will need to

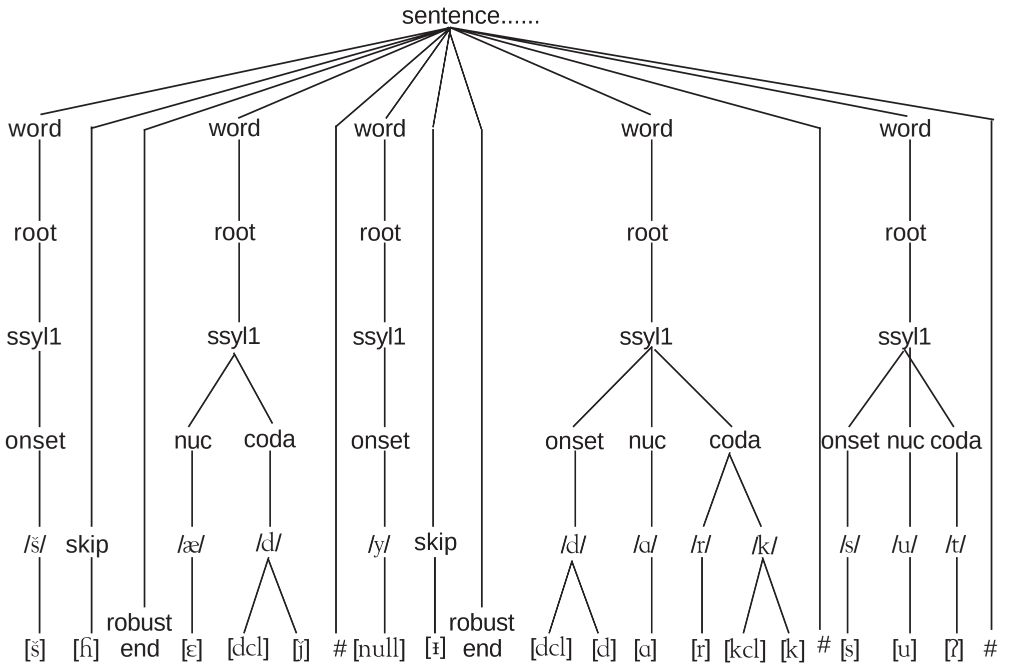


Figure 7-5: Example of the first half of an sa-1 sentence parse tree which contains skipped phones — “She had your dark suit...”

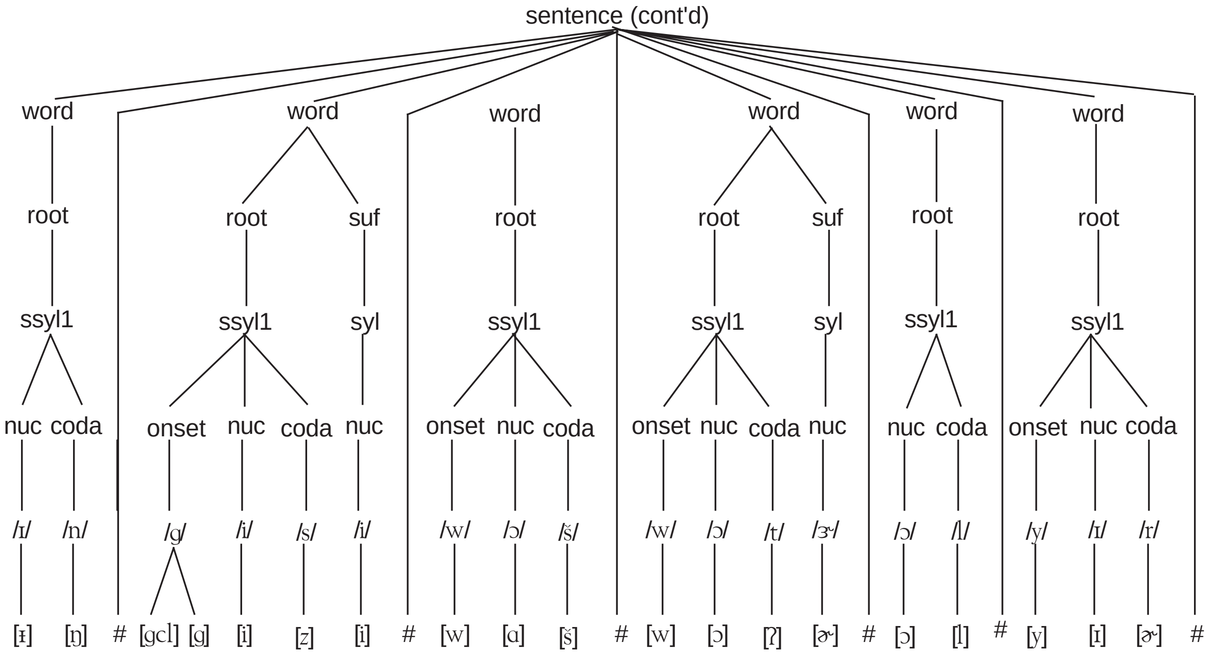


Figure 7-6: Example of the second half of sa-1 sentence parse tree — “...in greasy wash water all year.”

be explored further.

7.6.3 Lexical Access in the Extended Layered Bigrams

Lexical access in the extended layered bigrams is a straightforward table-lookup procedure. For each of the 21 words in the vocabulary, the table stores the spelling and a *single* phoneme sequence as the pronunciation. Given a complete sentence parse tree, lexical access involves extracting the phoneme sequences between adjacent WORD-END or ROBUST-END nodes, and mapping each phoneme sequence into a word spelling. For example, referring to Figure 7-5, the first three phoneme sequences are /š/, /æ d/ and /y/ respectively, none of which can map to any word. The fourth phoneme sequence, /dɑrk/, maps to "dark," and so forth. Therefore, lexical access based on this parse tree outputs the word sequence "NIL NIL NIL dark suit in greasy wash water all year."

Analysis of the training data reveals that under certain circumstances, the skipped phones should be incorporated into the parse. For example, in Figure 7-5 the skipped phone [ɦ] precedes a robust-end node. If this phone is incorporated into the sub-tree corresponding to the subsequent word, we should be able to salvage a correct parse for the following word "had." Therefore we have included a "second pass" in our parsing algorithm, which scans the complete parse tree of a sentence, and tries to incorporate the skipped phones into the subsequent word sub-trees. The output word sequence now becomes "NIL had NIL dark suit in greasy wash water all year," and the word accuracy has risen from 72.7% (8 out of 11 words) to 81.8% (9 out of 11 words).

7.7 Captured Phonological Variations

The phonological variations captured using the layered bigrams can be observed by examining the trained probabilities. Three types of variations have been brought into consideration — (i) allophonic variations, referring to the different realizations of a phoneme in the context of the neighboring phones/phonemes, (ii) across-word phonological variations, which account for the changes in a word pronunciation depending on the neighboring words, and (iii) within-word phonological variations, which contributes to alternative pronunciations for words.

7.7.1 Allophonic Variations

The different allophonic variations of a phoneme can be found by the enumeration of column histories. For example, the phoneme /t/ can be found in the words "suit," "water," "don't" and "that," in the context of {WORD ROOT SSYL1 CODA} — the coda of a stressed syllable in a root morph. According to our training data, this phoneme can be realized as several different phones — [ʔ] [ʕ], [d□] [d], [t□], [t] and [null]. The number of occurrences in each case is tabulated in Figure 7-7. In particular, the number of [d□]'s and [d]'s are the same because they both come from the same instances of released [d]'s. The number of [t□]'s is much higher than the number of [t]'s because there are abundant unreleased [t]'s in the training data. Certain allophones tend to dominate in particular contexts of neighboring phonemes or phones. This will be addressed in the next two subsections.

7.7.2 Across-word Phonological Variations

Across-word phonological variations refer to the allophonic changes that take place across a word boundary. Recall that in the layered bigrams, the prediction probability for the phoneme-phone pair towards the right of a word boundary is conditioned on the phoneme-phone pair to the left. This is based on the assumption that across a word boundary, the realization of the right-phoneme is dependent only on the left-phoneme and its realization. The advantage of making this assumption is that we can share training data across the left phoneme-phone pairs with different histories.[5]

[5] An implicit WORD-END condition is realized since the pooling is only done at word boundaries.

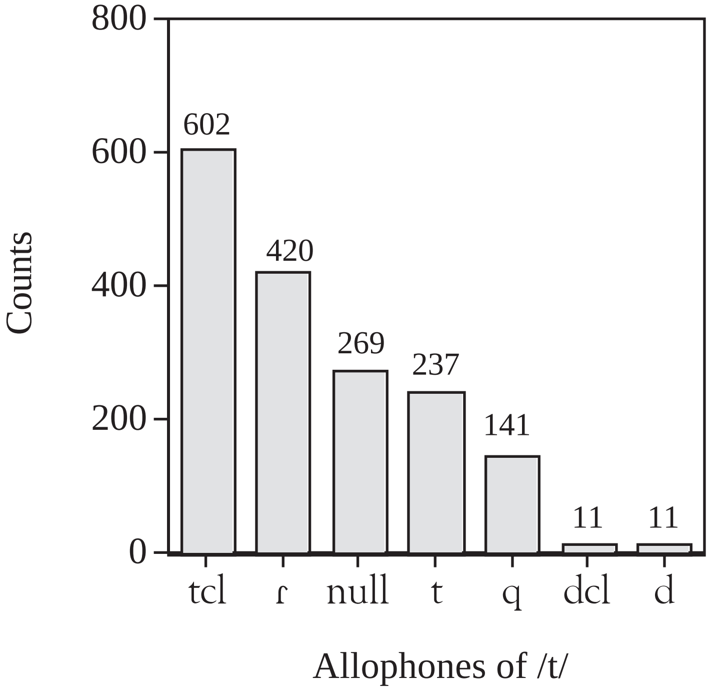


Figure 7-7: Bar graph showing the occurrences of the different allophones of /t/.

left-phone	# occurrences of the left context	right-phone
[null]	1	[y]
[ʕ]	3	[y]
[d□]	60	[y]
[d]	127	[y]
[č]	1	[null]
[ǰ]	170	[y] [null]

Table 7.1: Across-word phonological variations for the word sequence "had your."

By virtue of this assumption, across-word phonological variations can be observed by enumerating the different combinations of left phoneme, left phone, right phoneme and right phone. For example, at the word boundary between "had" and "your" in an "sa-1" sentence, we have found seven distinct combinations in our training data, as listed in Table 7.1.

The total number of occurrences of all left contexts is 362 in Table 7.1, equal to the number of training speakers and thus the number of training "sa-1" sentences which contain the specific word boundary. Three combinations in Table 7.1 show cases of *palatalization*, where the alveolar phoneme /d/ is realized as a palatal phone, i.e. [č] or [ǰ]. The single count of palatalization with [č] is followed by a deleted /y/. According to the trained probabilities, palatalization with /ǰ/ is followed by a deleted /y/ 96% of the time ($P_{across-word\ prediction}$(/y/-[null] | /d/-[ǰ]) = 0.96). There is strong indication in the probabilities that when /d/ is palatalized, the following /y/ is almost certain to be deleted.

7.7.3 Within-word Phonological Variations

The layered bigram probabilities also elicited within-word phonological variations, which may occasionally form alternate word pronunciations. These variations are exhibited by the distinct three tuples which correspond to a particular bottom-up

right-phoneme(RP)	right-phone(RS)	$P(RS\|LH)$
/ɑʸ/	[ʌ]	0.006
/ɑʸ/	[ɑ]	0.006
/ɑʸ/	[i]	0.002
/ɑʸ/	[æ]	0.002
/ɑʸ/	[ɑʸ]	0.984

LH = {WORD ROOT SSYL1 ONSET /l/ [l]}

Table 7.2: Within-word phonological variations for the word "like."

prediction probability. For example, bottom-up prediction of the phoneme /ɑʸ/ in the word "like" is conditioned on (i) the right sibling, which can be one of the five possible allophones of /ɑʸ/ in this context — [ʌ], [ɑ], [i], [æ] or [ɑʸ], and (ii) the left history, {WORD ROOT SSYL1 ONSET /l/ [l]}. This is summarized in Table 7.2. The table indicates that over 98% of the time, the phone [ɑʸ] succeeds the left-history {WORD ROOT SSYL1 ONSET /l/ [l]}. Evidently, the phoneme /ɑʸ/ in "like" is often distinctly pronounced.

Similarly, in the word "had," the distinct 3-tuples which correspond to the bottom-up prediction of /æ/ are summarized in Table 7.3. We see that the phoneme /h/ can be pronounced as the aspirant [h], a voiced aspirant [ɦ], or it can be deleted. The next vowel /æ/, is more often mapped to either [æ] or [ɛ]. The last case in the middle row in Table 7.3, where both /h/ and /æ/ are deleted, comes from a contraction where "She'd your..." was uttered instead of "She had your...." The statistics in Table 7.3 indicate that the tense vowel /æ/ in "had" is often realized as either the tense phone [æ] or the lax phone [ɛ]. If the left phoneme /h/ is either [h] or [null], the transition probabilities to [æ] and [ɛ] are roughly 0.5 and 0.4 respectively. However, if the left phoneme /h/ is voiced to become [ɦ], it becomes more probable for /æ/ to be pronounced as [æ] rather than [ɛ] (0.7 versus 0.3 probability). This may suggest that if the speaker is articulating carefully enough to voice the /h/, then it is more likely for the tenseness of the /æ/ to be preserved.

left-phone	# occurrences of left-history	right-phone(RS)	$P(RS\|LH)$
[h]	52	[æ]	0.560
		[ɛ]	0.400
		[e]	0.020
		[ɨ]	0.020
[null]	16	[æ]	0.50
		[ɛ]	0.440
		[null]	0.060
[ɦ]	294	[æ]	0.694
		[ɛ]	0.303
		[ɪ]	0.003

LH = {WORD ROOT SSYL1 ONSET /h/ *left-phone*}

Table 7.3: Within-word phonological variations for the word "had."

7.7.4 Capturing Phonological Rules

To illustrate how the layered bigrams capture a typical phonological rule, such as the *flapping* of /t/, we should examine and compare the words "don't," "suit," "water," as well as the word pairs "don't ask" and "suit in." In general, flapping of the phoneme /t/ occurs in an intervocalic position. Thus we should observe flapping in "suit" "water" and "suit in," but not "don't" or "don't ask."

In the word "don't" of an "sa-2" sentence, the phoneme /n/ is produced either as [n] or [n̩] in the training data (refer to Figure 7-3). The column {WORD ROOT SSYL1 CODA /n/ [n]} can advance to [null], [d□], [t□], [t], or [ʔ] with non-zero advancement probabilities ($P(RS \mid LH)$), but the advancement probability to the flap [ʕ] is zero.

The situation is different for "suit" and "water." The vowel /u/ in the word "suit" is realized as [ü], [ɨ] and [u] in the training data. Each of these phones can advance to [ʕ] with non-zero probability. Pooling the data across the three different left phones, the probability of flapping in "suit" following the left context of {WORD ROOT SSYL1 NUCLEUS /u/ } is about 0.09. If we inspect the across-word prediction probabilities with the *flapped* /t/ as the left context, i.e. $(P_{across-word\ prediction}(right\ phoneme -$ *phone* | /t/−[ʕ])), we see that the right-phoneme can only be /ɪ/, coming from the word sequence "suit in." Since the word "don't" does not terminate with the flap (i.e. the phone [ʕ]), the word pair "don't ask" does not contribute to the probability for flapping across a word boundary).[6]

Similarly, the different allophones corresponding to the vowel /ɔ/ in the word "water" include [ɑ], [ɔ], [ə], [r], [ʊ] and [ʌ], and each of these distinct left-histories can transition to a flap [ʕ] with non-zero probabilities. The overall probability of flapping in "water" following the context of {WORD ROOT SSYL1 NUCLEUS /ɔ/ } is about 0.32.

The gist of this subsection is to illustrate how the layered bigram probabilities encode phonological rules, using the *flapping* rule as an example. Flapping is prohibited for the /t/ in "don't" because the nasal in the left-history shows that the context is *not intervocalic*. The word pair "suit in" provides an *intervocalic*, *across-word* context, and therefore flapping is possible. The word "water" provides an *intervocalic*, *within-word* context and flapping is more likely. We believe that the probabilities used for parsing should help prioritize the applications of different phonological rules. This should be verified by conducting experiments using larger data sets.

7.8 Experimental Results

We measured both the word accuracies and sentence accuracies for our test set. A sentence is correct if lexical access yields a word sequence identical to the reference sentence. Figure 7-8 plots the performance values as the amount of training data is increased. As we increase the number of training speakers from 20 to 362 (the full set), the word accuracies for the "sa-1" and "sa-2" sentences rose by about 9% and 6% respectively, while the sentence accuracies increased by roughly 22% and

[6] Instead, the set of phonetic realizations for the phoneme /t/ in "don't ask" is composed of [t], [$t^{\square}$], [ʔ], [d] and [null].

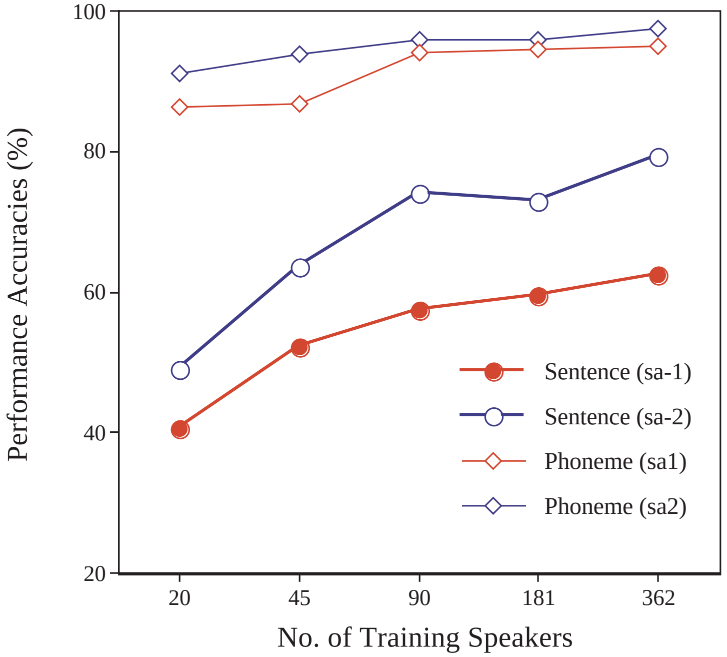


Figure 7-8: Word and sentence accuracies of the layered bigrams in parsing sentences, plotted as a function of increasing training data.

30% respectively. This indicates that the layered bigrams can capture more phonological variations from wider exposure to training data, leading to improvements in performance.

Most of the errors in the sentence parse trees are due to sloppy articulation in continuous speech. When the full training set is used, the mistakes made while testing the "sa-1" sentences, "She had your dark suit in greasy wash water all year," are exhaustively displayed in the following:

- The input phone sequence was:

  [š i ɦ æ d□ d y ɪ d□ d ɑ r k□ k s ü ɾ ɪ n g□ g r i s i w ɔ š w ɔ ɾ ɝ ɔ l y ɪ ɝ]

  Lexical access did not yield the word "your" due to the missing retroflexion.

- The input phone sequence was:

[š i æ d□ ǰ ɚ d□ d ɑ r k□ k s ü t□ ŋ̩ ɡ□ ɡ r i z i w ɑ š w ɔ ɾ ɝ $\underline{\text{ɔ}}$ y ɪ ɚ]

Error occurred with the word “all” due to the missing phone [l].

- The input phone sequence was:

  [š ɪ $\underline{\text{d□ ǰ ɨ}}$ d□ d ɑ r k□ k s ü ɾ ɨ ŋ ɡ□ ɡ r i s i w ɑ š w ɔ ɾ ɚ ɔ l y ɪ ɚ]

  The parser did not get the word “had” in the first complete parse tree popped off the stack, due to the absence of the phones [h] and [æ]. However, searching deeper into the stack showed that the second complete parse tree popped off would have inserted the phonemes and obtained the word “had.” The word “your” was also an error caused by the missing phones [y] and [r].

- The input phone sequence was :

  [š i ɦ æ d□ ǰ ɚ d□ d ɑ r k□ k s ü t□ ɪ n $\underline{\text{ɡ□ ɡ r ɪ z i}}$ w ɔ š w ɑ ɾ ɝ ɑ l y ɪ ɚ]

  The problem encountered with this utterance is with the word “greasy.” It was pronounced with the phone sequence [ɪz], which did not occur in the training data.

- The input phone sequence was:

  [š ɦ ɛ d□ $\underline{\text{ǰ ɨ}}$ d□ d ɑ r k□ k s ü ɾ ɨ ŋ ɡ□ ɡ r i z i w ɑ š w ɔ ɾ ɚ ɔ l y ɪ ɚ]

  This case is similar to the first example. The parser failed to recognize “your” due to missing retroflexion.

- The input phone sequence was:

  [š i ɦ æ d□ $\underline{\text{ǰ ü}}$ d□ d ɑ r k□ k s ü t□ t ɪ n ɡ□ ɡ r i s i w ɑ š w ɑ ɾ ɝ ɔ l y ɪ ɚ]

The occurrence of the vowel [ü] for the word "your" was not observed during training. Consequently the phone was skipped, leading to a word error for this utterance.

- The input phone sequence was:

  [š i ɦ æ d□ ǰɚ d□ d ɑ r k□ k s ü ɾ ɨ ŋ ɡ□ ɡ r i s iw ɑ š w ɔ ɾ ɝ ɔ l y ɨ ɝ]

  The problematic word here is "year," realized as the phone sequence [ɨɝ], and the probability going from the phone [ɨ] to [ɝ] is 0.

Similarly, mistakes made while testing the "sa-2" sentences - "Don't ask me to carry an oily rag like that," are as follows:

- The input phone sequence was:

  [d o n t□ t æ s k□ k m i ɾ ɨ k□ k ɛ r i n ɔʸ l i r æ ɡ□ ɡ l ɑʸ k□ k ð æ t□ t]

  The top-choice theory did not parse the word "in" correctly, based on the single phone [n]. The parse tree inserted the phonemes /h/ and /æ/ before /n/, mapping them both to [null] phones. However, /hæn/ did not map to a legitimate word according to the lexical access table-lookup procedure. The second best parse tree, however, did give the correct answer by inserting only the phoneme /æ/ before the /n/. The same error occurred in two other utterances:

  [d o n æ s m i ɾ ɨ k□ k ɛ r i n ɔʸ l i r æ ɡ□ ɡ l ɑʸ k□ ð æ t□]

  and

  [d o n̩ æ s m i ɾ ɨ k□ k ɛ r i n ɔʸ l ɨ r æ ɡ□ ɡ l ɑʸ k□ ð æ t□]

  Again, the top-choice hypotheses in each case missed the word "in," but the second-choice hypotheses recognized the correct word sequence.

- The input phone sequence was:

[d o ṇ æ s k$^{\square}$ m i ɾ i̲ k$^{\square}$ k ɛ r i ɨ ṇ o l i r æ ɡ$^{\square}$ ɡ l ɑ^{y} k$^{\square}$ k ð æ t$^{\square}$ θ]

The use of the vowel /i/ for the word "to" is unusual, and as a result the vowel was skipped in the parse tree, resulting in a word error.

- The input phone sequence was:

  [d o n æ s k$^{\square}$ i̲ ɾ ɨ k$^{\square}$ k ɛ r e ʔ æ n ɔ l i r æ ɡ$^{\square}$ ɡ l ɑ^{y} k$^{\square}$ ð æ ʔ]

  This phone sequence contains the rare omission of the phone [m] from the pronunciation of "me," which led to a word error.

It is perhaps worthwhile to recapitulate that our experimental results are based on the top-choice hypothesis from an inadmissible search procedure, and performance may potentially be improved if measures are taken to select amongst the top N hypotheses. Another possible source of improvement may be the provision of alternate phonetic choices with a recognizer, to take the place of the single phone sequence as input. Should the "top-choice" phone (based on the acoustics) be an uncommon occurrence (based on the layered bigram probabilities), the partial parse theory may still be able to survive by extending to other phones with lower acoustic rankings.[7] Consequently the parser should be less prone to fail or err.

7.9 Chapter Summary

This chapter describes a pilot study in which we attempt to demonstrate how the current research paradigm can be generalized to include other layers in the grand speech hierarchy. In particular, we have chosen to extend our the hierarchical lexical representation downwards to the phonetic level in order to capture phonological

[7] For example, it may happen that a reduced vowel in "like" is realized as [ə] and therefore this phoneme is most favored by the recognizer, over other options like [ʌ] and [ɑ]. However, according to Table 7.2, the advancement probability of the left-history {WORD ROOT SSYL1 ONSET /l/ [l] } to [ə] is zero. So if only the top-choice phone is provided for advancement, the partial theory will perish. But if the alternative choices are provided as well, the partial theory will survive.

rules. Our experiments utilize the "sa" sentences of the TIMIT corpus, and the layered bigrams are trained and then used to parse the phonetic transcriptions of the sa sentences. The parsing process is permissive of dialectal variations and coarticulatory effects like palatalization and deletion. The output word sequence is extracted from the top-choice parse theory. Within the scope of the small experimental data set, our results show that the layered bigrams can successfully capture a greater variety of phonological effects from an increasing amount of training data, bringing improvements to word and sentence accuracies. The extended layered bigrams can potentially be adapted for use in speech recognition, to analyze the phone lattice provided by the recognizer. A parse tree can be generated bottom-up, left-to-right based on the lattice, while the layered bigrams propose underlying phonemes for each phone it encounters, simultaneously enforcing the probabilistic phonological rules. Proceeding upwards, the layered bigrams can propose syllable parts, stress, and morphs, applying syllabification rules, stress and morphological constraints based on the probabilities.[8] A complete column history can then be advanced in its entire form to the next phone. In this way, the process of decoding the phone lattice, and the administration of lexical constraints can be elegantly united. The tedium of applying ordered rules thus becomes dispensable.

[8] This can be extended further upwards along the speech hierarchy, from the word level to syntactic and semantic constraints at the sentence level, from the sentence level to the discourse level, and so forth.

Chapter 8

Conclusions and Future Work

8.1 Thesis Summary

This thesis proposes a methodology for incorporating different linguistic knowledge sources into a common hierarchical framework for representing speech. This framework can potentially serve as a representation applicable to many areas of speech research, including recognition, synthesis and understanding. Having a unified approach for the different tasks not only minimizes redundancy in effort, but improvements in one particular task are inherited by the other tasks as well.

The feasibility of implementing this hierarchical representation for speech is demonstrated with a substructure of the grand hierarchy, on the test-bed of bi-directional letter-to-sound/sound-to-letter generation. We began with the design of a hierarchical lexical representation for the English word, as described in Chapter 2. Only a substructure of the grand hierarchy proposed for speech is included in the design as stratified linguistic representations — the generic English word unit, morphology, stress, syllabification, broad manner classes, phonemes and graphemes. These linguistic knowledge sources play an influential role in the determination of English word pronunciations in relation to the word spellings. Each level in the representation propagates its own set of linguistic constraints—morphological decomposition, stress contour, syllabic constraints such as Sonority Sequencing, the Maximum Onset Principle and Stress Resyllabification, phonotactics and graphotactics. Therefore, the lexical representation aggregates the descriptive power of variable-sized units, which range from coarse to fine as we descend the hierarchical parse tree.

Chapter 3 continues to describe the parsing algorithm used in conjuction with the lexical representation to achieve letter-to-sound and sound-to-letter generation. The idea is to combine the two tasks in a single parsing framework. An input spelling (or pronunciation) is analyzed to obtain a complete parse tree, from which the output pronunciation (or spelling) is derived. The parser, which is referred to as the "layered bigrams," is a hybrid of rules and statistics. A natural language parser, TINA, is used to generate training parse trees according to the linguistic markings of the training corpus and a set of hand-crafted, context-free rules. The training parse trees are then used for training the layered bigram probabilities. These probabilities capture not only the straightforward constraints specified by the context-free rules, but also the more subtle constraints that are automatically discovered from the training parse trees during the training procedure. Therefore, the constraints propagated along each layer in the hierarchical lexical representation, as well as the constraints governing the interactions between successive layers, are assimilated as probabilities.

During testing, the parser creates a parse tree based on the input. The various constraints are enforced within the lexical representation, and the parse tree is a complete analysis of the input word (spelling or pronunciation) at the seven lingusitic layers in the hierarchy. Many partial parse theories are produced and placed on a stack during parsing, and a best-first search strategy is used to determine the order in which the different partial theories are pursued. A longer theory tends to have a lower score than a shorter theory, because the former score includes more probabilities in the product. To facilitate the comparison of partial theories of different lengths, a "fading" evaluation criterion is used to "normalize" the stack score of each partial theory with respect to its length. The experimental results reported in Chapter 4 show that aside from a small proportion of nonparsable words in the development test set, the layered bigrams parser has attained competitive performance for generation in *both* directions.

The importance of the higher-level linguistic knowledge in our generation tasks is empirically verified in Chapter 5. Two comparative studies were conducted based on letter-to-sound generation — the first investigates the relative contribution of the different linguistic levels of representation towards generation accuracies, and the second examines the merits of the hierarchical framework in comparison with a non-linguistic approach.

In the first study, we discovered that as different layers are omitted from the training parse trees, linguistic constraints are lost, manifested as a decline in generation accuracy and an increase in perplexity and coverage. The converse is true when the layer of broad classes is omitted—generation accuracy was gained, while perplexity and coverage decreased. The exception of the broad class layer may be caused by the fact that the broad classes can be predicted from the phonemes bottom-up with certainty. Their inclusion may have led to excessive smoothing in the subsequent predictions in the upper levels of the hierarchy.

In the second study, the hierarchical parsing framework is compared with a *non-linguistic* approach. This alternative approach does not utilize any higher-level linguistic knowledge such as morphology, stress or syllabification. Instead, it requires a one-to-one letter-to-phoneme mapping for each training word as *a priori* information. The training procedure creates a record of all possible word "fragments"(up to a pre-set maximum fragment length)—each containing a partial word spelling, the corresponding partial transcription, a bigram language score and a phonemic transcription score. Generation involves a Viterbi search across all the possible ways in which the spelling of the test word can be reconstructed from the partial spellings in the word fragments. Therefore, contextual information is captured within the word fragments, but no higher level linguistic knowledge is explicitly used. It was observed that the two different approaches to letter-to-sound generation gave comparable performance, but the hierarchical approach requires 20 times fewer parameters for bi-directional generation, when compared to the number of parameters required by the non-linguistic approach for uni-directional generation. This indicates that the use of higher-level linguistic knowledge helps impose structure on the lexical representation, which becomes a more parsimonious description of English graphemic-phonemic mappings.

In Chapter 6 we proceed to investigate how constraints can be relaxed in the layered bigrams framework to widen the parser's coverage for nonparsable words. The layer of broad classes is omitted from the hierarchical lexical representation for the subsequent experiments, since we have discovered that this omission leads to slight performance improvement. Analysis of the nonparsable words in our previous experiments has shown that they are the outcome of zero left-to-right advancement probabilities, which mainly arise in the context of compound words, words with geminate letters, and words with sparse data problems. Consequently, a robust parsing strategy is targeted at handling these three situations. Compound words are composed of one or more individual words in tandem to become a new distinct word. Since data is most sparse at the boundaries of concatenation, a natural way to parse a compound word is to allow a partial parse to end at the word boundary in the compound, and start another parse to account for the next word(s). Geminate letters which are pronounced as a single consonant constitute a single graphemic terminal in the corresponding parse tree of the word. These geminate letter terminals tend to have rarer occurrences than their single letter counterparts, but their pronunciations are often identical. Therefore, if we allow relaxation of the graphemic constraints to replace a geminate letter terminal with the corresponding single letter terminal, we can often resolve nonparsability and produce a reasonable phonemic pronunciation. As regards the rest of the nonparsable words which have sparse data problems, we offer the solution of sharing probabilities amongst similar left contexts, by conditioning the left-to-right advancement probabilities on the *partial* left-history in place of the *entire* left-history.

This robust parsing strategy is invoked as a backup mechanism for the original parser. By so doing we can minimize the number of partial theories explored. Experimentation with this configuration yielded coverage and slight performance improvements for both the development test set and the real test set. This set of experiments serves to illustrate how generation constraints can be easily relaxed within the probabilistic parsing framework to attain better coverage of the data.

Chapter 7 describes a pilot study which demonstrates the generalizability of the layered bigrams parsing paradigm to other linguistic levels in the hierarchy. In particular, a layer of phones is added to the hierarchical lexical representation. The layered bigrams are also extended so that phonological rules are captured in terms of probabilities between the layers of phonemes and phones. The scope of this pilot study is restricted to the two "sa" sentences in the TIMIT corpus, which are especially designed for the study of phonological and dialectal variations. The layered bigrams generate a parse tree from the phonetic transcription of an entire sentence, and from the tree we extract the sequence of spoken words. Results indicate that as the amount of training data increases, a greater diversity of *within-* and *across*-word phonological variations is observed and captured in the layered bigram probabilities, leading to marked improvements in sentence and word accuracies. The results of this preliminary study are promising, but due to the limited amount of data involved, further experimentation is necessary.

The ultimate goal of this thesis is to propose a grand speech hierarchy, which incorporates a wealth of linguistic knowledge, to be used as a common framework for speech synthesis, recognition and understanding. The work presented only represents an initial attempt in utilizing the proposed framework for letter-to-sound and sound-to-letter generation. This work can be further pursued in a number of future directions. Several ideas are offered in the following sections.

8.2 Performance Improvement

The contextual information utilized by the layered bigram parser for generation is limited only to the left history. There is much room for performance improvement as more elaborate contextual information is considered. The parsing procedure can be modified so as to carry richer context for incremental prediction. Alternatively, the layered bigrams can be viewed as an inexpensive means for generating *multiple* plausible hypotheses, and refinement processes can follow. The refinement can exploit various kinds of "future context" — letters, phonemes, syllables, suffixes, etc., which is considered by some to be more important than the left context [3][1], [36]. The combination of left *and* right contexts is vital for tracing stress dependencies, which are known to spread over a long range of several syllables [12] and thus cannot be determined locally. Therefore post-processes can use additional contextual information to filter the hypotheses, select the most desirable one from the pool, or correct systematic generation errors. We have made a preliminary trial attempt to design a post-process of such ilk using a typical induction technique known as "transformation-based error-driven learning." This is a learning algorithm previously used for part-of-speech tagging [9]. In this work the learning algorithm is used for the automatic inference of refinement rules for generated pronunciations.

The learning algorithm trains on two sets of letter-to-phoneme alignments — the "correct" alignments from the layered bigrams training parse trees, and the "generated" alignments from the layered bigrams output.[2] The translation of a training parse tree into a letter-to-phoneme alignment follows a simple convention. If a phoneme in the training parse tree maps to a grapheme of more than one letter, then

[1] The first of three passes in MITalk's letter-to-sound rules strips suffixes in a right to left direction.

[2] This is the top-choice pronunciation output for each training word, with the layered bigrams trained on the full training set.

the first letter will claim the phoneme, and the rest will get the /null/ phoneme. Consonant phonemes which map to geminate letters[3] in the parse tree are treated slightly differently. Both letters will get the same phoneme in producing the one-to-one letter-to-phoneme alignment. For example, the "correct" letter-to-phoneme alignment for the word "blocked" is:[4]

B	L	O	C	K	E	D
/b/	/l/	/ɑ/	/k/	/k/	/t/	/null/

The generated parse tree for "blocked" does not have the correct pronunciation. The generated alignment is:

B	L	O	C	K	E	D
/b/	/l/	/ɑ/	/k/	/k/	/ᵻ/	/d/

The learning algorithm seeks to learn refinement rules to correct the generated alignments. The procedure is summarized in the following steps:

1. Compare the "current" and "correct" alignments and record all errors. The "current" alignments are initialized as the layered bigram alignments.

2. Propose refinement rules based on a set of rule templates and the errors observed. The rule templates are shown in Table 8.1. These rules look beyond the left column history, to include context up to three phonemes/letters to the left/right of the current phoneme.

[3] This also applies to the letter sequence "ck" even though it is not a geminate. This is because the letters 'c' and 'k' are often pronounced as the phoneme /k/, similar to two geminate letters being pronounced as the same phoneme.

[4] In cases where more than one letter is mapped to a phoneme, the alignment follows the convention that the first letter gets mapped to the phoneme, and the other letters are mapped to /null/.

Change P_0 from /A/ to /B/ if P_{-1} = /C/ and P_1 = /D/.
Change P_0 from /A/ to /B/ if P_{-2} = /C/ and P_{-1} = /D/.
Change P_0 from /A/ to /B/ if L_0 = c and L_2 = d.
Change P_0 from /A/ to /B/ if L_1 = c.
Change P_0 from /A/ to /B/ if L_0 = c and P_2 = /D/.
Change P_0 from /A/ to /B/ if P_1 or P_2 is /C/.
Change P_0 from /A/ to /B/ if P_1 or P_2 or P_3 is /C/.
Change P_0 from /A/ to /B/ if L_1 = c and L_2 = d.
Change P_0 from /A/ to /B/ if P_1 = /C/ and P_2 = /D/.
Change P_0 from /A/ to /B/ if L_{-1} = c and L_{-2} = d.
Change P_0 from /A/ to /B/ if P_{-1} = /C/.
Change P_0 from /A/ to /B/ if L_1 or L_2 is c.
Change P_0 from /A/ to /B/ if L_0 = c and P_{-1} is /D/.
Change P_0 from /A/ to /B/ if P_{-1} or P_{-2} or P_{-3} is /C/.
Change P_0 from /A/ to /B/ if L_0 = c and P_2 = /D/.
Change P_0 from /A/ to /B/ if P_1 = /C/.
Change P_0 from /A/ to /B/ if P_{-1} = /C/ and P_{-2} = /D/.

Table 8.1: Some examples of rule templates for transformational error-driven learning. These rules include context up to a window of seven phonemes/letters centered at the current phoneme/letter, i.e. the windows are $P_{-3}P_{-2}P_{-1}P_0P_1P_2P_3$ and $L_{-3}L_{-2}L_{-1}L_0L_1L_2L_3$, where P_0 is the current phoneme, and L_0 is the current letter.

3. The refinement rule which brings about the maximum number of corrections is learnt and used to update all the current alignments.

4. The process repeats until the incremental improvement drops below a threshold.

A total of 86 rules are learnt based on the training data. Their application to the generated outputs on the development test set brought the word accuracy from 71.8% to 73.5%, and the phoneme accuracy from 92.5% to 93.1%. In particular, two rules are used to correct the pronunciation error for “blocked”. Recall that the initial letter-to-phoneme alignment is:

B-/b/ L-/l/ O-/ɑ/ C-/k/ K-/k/ E-/ɪ/ D-/d/

Application of the first rule:

Change P_0 from /ɪ/ to /t/ if P_{-1} = /k/ and P_1 = /d/.

results in the intermediate alignment:

B-/b/ L-/l/ O-/ɑ/ C-/k/ K-/k/ E-$\underline{\text{/t/}}$ D-/d/

Application of the second rule:

Change P_0 from /d/ to /null/ if P_{-2} = /k/ and P_{-1} = /t/.

gives the correct pronunciation in the alignment:

B-/b/ L-/l/ O-/ɑ/ C-/k/ K-/k/ E-/t/ D-$\underline{\text{/null/}}$

Similar results have been obtained by Huang et al. [37]. Their experiments adopted a similar learning algorithm (with similar rule templates) for letter-to-sound generation with 3,600 training words and 425 testing words. The experimental corpora consist of the CMU Pronunciation Dictionary [84] and the high-frequency words in the Brown Corpus. The initial alignments for the training procedure are obtained by mapping each letter in a word spelling to its most frequent phoneme. A total of 580 rules were learnt and a phoneme accuracy of 87.3% was achieved.[5]

In addition to the neighboring letters and phonemes, other contextual information, such as part of speech, stress contour filters, morph filters, and spelling change rules or spell checkers, etc. should also be propitious for automatic letter-to-sound and sound-to-letter generation. Much work can potentially be done in developing post-processes to refine generation outputs.

8.3 Large Vocabulary Speech Recognition

The framework proposed in this thesis should be suitable for large-vocabulary speech recognition in a variety of ways. The lexical representation is an attractive candidate for large-vocabulary tasks because it has the advantage of extensive structural sharing

[5] Although the two experiments should not be strictly compared due to the use of different data sets, the deviation in performance accuracies may suggest that certain constraints captured in the layered bigrams are not acquired by using the learning algorithm alone.

among words which should lower storage requirements. Words in a highly inflected language like English can be collapsed together according to similar morphological compositions. Allen has estimated a savings factor of 10 if a lexicon stores morphemes instead of all possible forms of words [2] [13]. Suhm et al [82] performed a study on the orthographic transcriptions in the Wall Street Journal (WSJ) domain. They classified more than 1,000 words that lie outside a known vocabulary of approximately 14,000 words, and found that 45% were inflections of in-vocabulary words, and 6% were concatenations of in-vocabulary words. The hierarchical lexical representation can be extended through the phone level to the acoustic level, possibly by way of phonetic classification to bridge the mapping between the discrete set of phones and the continuously varying acoustics. Bottom-up, left-to-right processing in the layered bigrams can tightly couple lexical constraints with the search algorithm in the phonetic recognizer. The layer of broad manner classes can be used for rapid lexical access (fast match) by narrowing down possible word candidates to a short list which belong to the same cohort [75]. Probabilistic phonological rules captured between the phoneme layer and the phone layer can offer alternate word pronunciations for decoding, and this should be intrinsic because the probabilities belong to part of a coherent whole in the layered bigrams framework. Automatic bi-directional letter-to-sound/sound-to-letter generation can be useful for tackling the out-of-vocabulary problem in speech recognition. A spoken language system cannot be expected to be able to fully specify its active vocabulary based on a static initial set. In the event that an out-of-vocabulary word is detected by the speech recognizer, the corresponding phone subsequence or subnetwork can be extracted from the phonetic recognizer, from which possible spellings can be generated. These spellings can then be verified against dictionaries, databases or directories. Alternatively, pronunciations can be generated when given the spellings of new words. The vocabulary of a large-vocabulary system can subsequently be dynamically updated with the generated spellings/pronunciations.

8.4 Interface with Pen-based Systems

The layered-bigrams paradigm, being probabilistic in nature, can accomodate uncertainty in the input. Therefore, it is a possible means of interfacing speech-based systems with pen-based systems, such as constructing handwriting-to-speech generators. The outputs from a handwriting recognition system (or optical character recognition system) can be channelled along with their respective probabilities to become the input of a letter-to-sound generation system. The top-choice pronunciation corresponding to a written or scanned input can be selected based on the *combined* scores of the handwriting system and the layered bigrams. A slightly modified application for pen-based systems is to use the layered bigrams as a language model to guide "letter-pair" prediction with lexical constraints. The effectiveness of the layered bigrams as a predictor of the next letter/character is roughly estimated by measuring the perplexity per letter/character of the development test set. We obtained a perplexity of 8.3 which is more constraining than a standard bigram language model (perplexity = 11.3), and comparable with a standard trigram language model (perplexity = 8.3).

8.5 Multilingual Applications

It should be possible to apply our system methodologies to multilingual systems whenever the letter-sound correspondences and context interact in the same way[6] as our current monolingual system [27], e.g. generating English name pronunciations in terms of the Japanese Katakana pronunciation alphabet. Some resemblances can be found between our formalism and the Speech Maker Formalism for Dutch [89]. Speech

[6] This is, of course, dependent on the language. Some languages, such as Chinese, do not lend themselves easily to translating graphemes into phonemes. Some other languages may have a close fit between its graphemic and phonemic forms, e.g. Spanish can be thoroughly characterized by approximately 35 to 40 letter-to-sound rules [90]. English, in comparison, is much more complicated, due to numerous loan words that are not anglicized.

Maker is used for text-to-speech synthesis, and supports a multi-level synchronized data structure called a *grid*. The grid contains *streams*, or levels which contain information about word accent, word class, and the morphemes, syllables, phonemes and graphemes constituting the word. Each stream is also synchronized with other streams in the grid by sync marks, i.e. the vertical bars as illustrated in Figure 8-1, which shows a portion of a grid. Generation in the Speech Maker is achieved by complex, *two-dimensional* rules which can operate on more than one stream at a time, modelled after the DELTA language [34]. An example is shown in Figure 8-2, which is a rule stating that an “A”, followed by any sequence of characters pronounced as a single consonant, followed by an “E”, which is root final, should be pronounced as /e/.

The rules in our hybrid approach differ from the Speech Maker in that they are less cumbersome. Each rule in the small rule set characterizes the derivations in a single linguistic layer, and is later converted into probabilities. The probabilistic parser utilizes multiple levels in the hierarchical lexical representation simultaneously during parsing, as it advances each column in the parse tree uniformly from left to right. By virtue of this, sync marks are unnecessary in our framework.[7] The set of probabilities used in the parser are conditioned upon an elaborate but easily specified left context, the entire left column history, which provides lingustic constraints for generation. Consequently, complex rules are inessential provided that we have sufficient data for training.

[7] Some may consider that sync marks are implicit in our framework, because the bottom-up prediction process identifies the locations where the predicted category merges with previously generated structures.

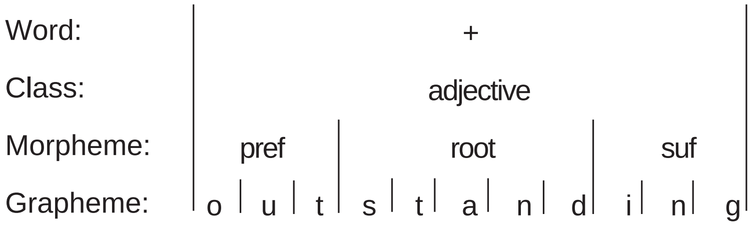


Figure 8-1: A portion of the Speech Maker grid representing the word “outstanding.”

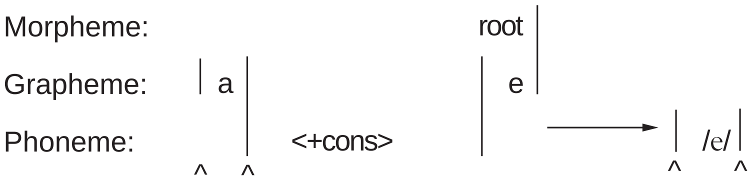


Figure 8-2: An example of a two-dimensional rule in Speech Maker. The upward arrows delineate the letter to be transcribed and the corresponding phoneme. The rule expresses that the letter “a” which precedes an arbitrary number of consonants and ending with the letter “e” should be pronounced as /e/.

8.6 Speech Generation, Understanding and Learning in a Single Framework

As was previously mentioned, the dimensions of the hierarchical representation can potentially be extended upwards to encompass natural language constraints [74], prosody, discourse and even perhaps dialogue constraints, and augmented downwards to include a layer capturing phonetics and acoustics. The full speech hierarchy can provide a common framework for speech generation, understanding and learning. This thesis has mainly covered the facet of generation. Understanding may be achieved because semantic and syntactic information can be extracted from the morphology layer. New words may be learnt and their regular inflectional and derivational forms can be automatically generated in the layered bigrams framework. There is ample room for further development in the quest for an integrated framework for speech generation, understanding and learning.

Appendix A

List of Morphs

PREF (i.e., prefix)

ROOT

ROOT2 (i.e., second unstressed syllable of the root)

SUF (i.e., suffix)

JOIN (denotes the concatenation of words into compound words)

Appendix B

List of Syllables

SSYL1 (i.e., syllable carrying primary stress)

SSYL2 (i.e., syllable carrying secondary stress)

SYL (i.e., unstressed syllable)

JOIN-SSYL1 (i.e., syllable containing the "connecting vowel" which carries primary stress, e.g. "acceler<u>o</u>meters")

JOIN-SYL (i.e., syllable containing the "connecting vowel" which is reduced, e.g. "hor<u>i</u>zontal")

ISUF (i.e., inflectional suffix, which is always an unstressed syllable)

DSUF (i.e., derivational suffix)

Appendix C

List of Subsyllabic Units

ONSET

NUC (i.e., nucleus)

CODA

M-ONSET (i.e., "moved" onset)

JOIN-VOW (i.e., "connecting vowel")

Appendix D

List of Broad Manner Classes

AFFR (i.e., affricate)

FRIC (i.e., fricative)

SEMI (i.e., semi-vowel)

VOW (i.e., vowel)

ASPIRANT

NASAL

STOP

Appendix E

List of Phonemes

This appendix lists the set of phonemes used in this thesis. There are 52 "phonemes" in total.

/h/	"ham"	/i/ (stressed)	"creed"
/č/	"church"	/i/ (unstressed)	"city"
/ǰ/	"enjoy"	/ɪ/	"bits"
/m/	"madame"	/e/	"brave"
/n/	"name"	/ɛ/	"bread"
/ŋ/	"planning"	/æ/	"cab"
/m̩/	"realism"	/ʌ/	"club"
/w/	"wage"	/ɑ/	"cobb"
/l/	"lyrics"	/ɔ/	"crawl"
/y/	"youth"	/o/ (stressed)	"crowd"
/r/	"run"	/o/ (unstressed)	"domain"
/l̩/	"rifle"	/u/ (stressed)	"cool"
/b/	"baby"	/u/ (unstressed)	"superbly"

/d/	"daily"	/ʊ/	"took"
/g/	"gain"	/ᵻ/ (/ᵻ/ and /ə/ collapsed)	"tilted"
/p/	"pace"	/ɑ^{w}/	"towns"
/t/	"table"	/ɑ^{y}/	"tribes"
/k/	"cookie"	/ɔ^{y}/	"voice"
/f/	"fabric"	/ɝ/	"actor"
/v/	"vague"	/ɔ r/	"port"
/θ/	"wealth"	/ɪ r/	"fear"
/ð/	"this"	/ɛ r/	"air"
/s/	"this"	/ɑ r/	"arm"
/z/	"zone"	/y u/	"vue"
/š/	"wish"	/ɑ l/	"salt"
/ž/	"regime"	/o l/	"polls"

Appendix F

List of Graphemes

This appendix lists the set of graphemes used in this thesis. There are 205 graphemes in total.

- *Regular* graphemes:
  #[you] #[wr] #[eh] #[wo] #[cc] #[eir] #[aigh] #[irr] #[eare] #[sch] #[ah] #[rh] #[uer] #[pt] #[ps] #[ayer] #[erre] #[qu] #[ort] #[eur] #[ju] #[eu] #[eo] #[uor] #[ieu] #[awr] #[aur] #[fe] #[oh] #[is] #[on] #[iew] #[ugh] #[owe] #[hou] #[ho] #[ole] #[ste] #[ire] #[gu] #[eor] #[oe] #[aul] #[io] #[eye] #[gh] #[wh] #[eigh] #[oub] #[ot] #[ut] #[eb] #[urr] #[oll] #[mp] #[ei] #[mm] #[olo] #[gne] #[oal] #[aire] #[st] #[tte] #[augh] #[gue] #[ia] #[pe] #[ye] #[zz] #[uy] #[yr] #[tch] #[ddh] #[ui] #[igh] #[dg] #[dt] #[our] #[orr] #[mb] #[bb] #[ff] #[ew] #[che] #[lle] #[err] #[gg] #[eer] #[eau] #[the] #[rr] #[arr] #[ore] #[et] #[be] #[aw] #[are] #[mn] #[au] #[ck] #[ere] #[ph] #[gn] #[me] #[ii] #[as] #[ue] #[pp] #[ear] #[que] #[wer] #[oy] #[nne] #[nn] #[z] #[ze] #[ay] #[ough] #[al] #[ey] #[ll] #[all] #[ke] #[th] #[ol] #[k] #[ne] #[de] #[ea] #[ee] #[ar] #[w] #[oo] #[x] #[air] #[f] #[aer] #[ae] #[sc] #[tt] #[ir] #[oi] #[j] #[h] #[dd] #[ge] #[re] #[ai] #[q] #[dge] #[ow] #[kn] #[v] #[ie] #[ch] #[*] #[ss] #[ng] #[er] #[se]

#[ur] #[c] #[te] #[ce] #[s] #[p] #[u] #[oa] #[r] #[ve] #[ou] #[g] #[or] #[sh] #[oar] #[le] #[y] #[es] #[l] #[i] #[t] #[e] #[m] #[ed] #[o] #[d] #[n] #[b] #[a]

- *Underbar* graphemes:

  #[ar_e] #[k_e] #[th_e] #[p_e] #[c_e] #[n_e] #[z_e] #[g_e] #[v_e] #[s_e] #[r_e] #[m_e] #[d_e] #[t_e] #[l_e] #[b_e]

Appendix G

Context-free Rules

This appendix lists the context-free rules used to generate the training parse trees.

WORD → [PRE] ROOT [SUF]

WORD → ROOT [SUF] JOIN ROOT

WORD → PRE JOIN ROOT

WORD → [PRE] ROOT ROOT-MARKER ROOT2 [SUF]

WORD → ROOT ROOT-MARKER ROOT2 [SUF] JOIN ROOT

JOIN → COMPOUND-MARKER [JOIN-SSYL1]

JOIN → COMPOUND-MARKER [JOIN-SYL]

JOIN-SYL → [M-ONSET-MARKER] [M-ONSET] JOIN-VOW

JOIN-SSYL1 → [M-ONSET-MARKER] [SSYL1-MARKER] JOIN-VOW

JOIN-VOW → VOW

ROOT → (SSYL1 SYL) (SSYL1 SYL)

ROOT → SSYL2

ROOT2 → (SSYL1 SSYL2 SYL)

SSYL1 → [ONSET] SSYL1-MARKER NUC [CODA]

SSYL1 → M-ONSET-MARKER M-ONSET SSYL2-MARKER NUC

SYL → [ONSET] NUC [CODA]

SYL → M-ONSET-MARKER M-ONSET NUC

M-ONSET → [FRIC] [STOP] SEMI

M-ONSET → FRIC

M-ONSET → NASAL

M-ONSET → STOP

M-ONSET → AFFR

ISUF → [NUC] CODA

ISUF → NUC

ISUF → M-ONSET-MARKER M-ONSET NUC

DSUF → CODA

PRE → (SSYL1 SSYL2 SYL) PRE-MARKER (SSYL1 SSYL2 SYL) PRE-MARKER

SUF → DSUF-MARKER (SSYL1 SSYL2 SYL DSUF) DSUF-MARKER (SSYL1 SSYL2 SYL)

SUF → DSUF-MARKER (SSYL1 SSYL2 SYL DSUF)

SUF → DSUF-MARKER (SSYL1 SSYL2 SYL DSUF) ISUF-MARKER ISUF

SUF → ISUF-MARKER ISUF

SUF → ISUF-MARKER ISUF ISUF-MARKER ISUF

ONSET → [ASPIRANT] [NASAL] [FRIC] [STOP] SEMI

ONSET → FRIC [FRIC]

ONSET → FRIC [NASAL]

ONSET → FRIC [SEMI]

ONSET → STOP [SEMI]

ONSET → (NASAL AFFR)

ONSET → (ASPIRANT

NUC → VOW

CODA → [NASAL] (STOP FRIC AFFR)

CODA → (NASAL FRIC AFFR)

CODA → (NASAL FRIC) STOP

CODA → [STOP] (FRIC) STOP)

CODA → SEMI (FRIC) AFFR NASAL STOP)

FRIC → (/f/ /v/ /θ/ /ð/ /s/ /z/ /š/ /ž/)

STOP → (/p/ /t/ /k/ /b/ /d/ /g/)

AFFR → (/č/ /ǰ/)

NASAL → (/m/ /n/ /ŋ/)

SEMI → (/w/ /r/ /l/ /y/)

VOW → (/ɪ/ /ɛ/ /æ/ /ɑ/ /ɔ/ /ʌ/ /ʊ/ /ɝ/ /l̩/ /m̩/ /ɑʷ/ /o/ /ɔʸ/ /ɑʸ/ /e/ /i/

/u/ /ɨ/ /y u/ /ɛ r/ /ɔ r/ /ɪ r/ /ɑ l/ /o l/ /ɑ r/ /i/(unstressed) /u/(unstressed) /o/(unstressed))

ASPIRANT → (/h/)

/w/ → (#[u] #[a] #[w] #[wh] #[o] #[we] #[*] #[ugh] #[e] #[i] #[ub] #[l] #[t] #[ju])

/l/ → (#[l] #[ll] #[le] #[lle] #[l_e])

/l̩/ → (#[al] #[l] #[le] #[l_e])

/h/ → (#[h] #[wh])

/r/ → (#[r] #[re] #[rr] #[r_e] #[rh] #[wr])

/y/ → (#[*] #[i] #[e] #[y] #[a] #[u] #[t] #[ea] #[gh] #[igh] #[ou])

/ɪ/ → (#[i] #[e] #[a] #[y] #[*] #[ae] #[ea] #[ai] #[ee] #[ui] #[u] #[ia] #[hi] #[ie] #[ei] #[o])

/ɨ/ → (#[u] #[e] #[y] #[a] #[au] #[i] #[o] #[ai] #[*] #[ae] #[on] #[ou] #[ah] #[ea] #[ough] #[ol] #[ei] #[ui] #[io] #[ia] #[eo])

/ɛ/ → (#[a] #[e] #[ae] #[ai] #[ea] #[u] #[eb] #[eh] #[ei] #[ie] #[oe] #[ay])

/æ/ → (#[a] #[o] #[ae] #[al] #[ho] #[au])

/ɑ/ → (#[o] #[augh] #[a] #[ow] #[i] #[ah] #[y] #[u] #[al] #[e] #[eye] #[ea] #[ho] #[oh] #[a])

/ɑ r/ → (#[ar] #[arr])

/ʌ/ → (#[e] #[u] #[o] #[ou] #[a] #[*] #[ia] #[oo] #[ai] #[io] #[iou] #[ae] #[ol] #[on] #[ah] #[ough])

/ɔ/ → (#[ou] #[aw] #[ough] #[o] #[ao] #[oa] #[a] #[au] #[as] #[augh] #[hau] #[aul] #[eo] #[al])

/ʊ/ → (#[oo] #[*] #[o] #[u] #[ou])

/ɝ/ → (#[or] #[ur] #[er] #[re] #[ir] #[r] #[ar] #[wer] #[our] #[yr] #[olo] #[urr] #[ear] #[ire] #[err] #[ure] #[uor] #[eur] #[irr] #[orr])

/ɑʷ/ → (#[ou] #[ow] #[aw] #[oub] #[hou] #[owe])

/o/ → (#[o] #[ough] #[oa] #[ow] #[ou] #[eau] #[ot] #[oe] #[ol] #[oh] #[owe] #[ew] #[o])

unstressed /o/ → (#[o] #[ow] #[eau])

/o l/ → (#[ol] #[oal] #[oll] #[ole])

/ɑ l/ → (#[ol] #[all] #[al])

/ɔʸ/ → (#[oy] #[oi])

/ɑʸ/ → (#[i] #[y] #[eye] #[igh] #[eigh] #[ui] #[ie] #[ye] #[uy] #[is] #[ei])

/e/ → (#[ay] #[a] #[ai] #[ey] #[eigh] #[au] #[eh] #[ea] #[e] #[ee] #[et] #[ae]

#[ei] #[aigh])

/i/ → (#[i] #[y] #[e] #[ie] #[ee] #[ey] #[ae] #[ea] #[is] #[ei] #[ii] #[ay] #[eo] #[oe] #[eh])

unstressed /i/ → (#[i] #[y] #[e] #[ey] #[ie] #[ee] #[ay])

/u/ → (#[u] #[oo] #[o] #[ew] #[ui] #[ue] #[ou] #[ieu] #[eu] #[wo])

unstressed /u/ → (#[u] #[ew] #[ou] #[ieu] #[o])

/y u/ → (#[u] #[eau] #[ew] #[ugh] #[ou] #[ut] #[ue] #[iew] #[eu] #[you])

/ɛ r/ → (#[aer] #[air] #[ar] #[er] #[ere] #[are] #[arr] #[ear] #[err] #[ur] #[aire] #[erre] #[ayer] #[ar_e] #[eir])

/ɔ r/ → (#[oar] #[or] #[ar] #[ore] #[orr] #[our] #[eor] #[aur] #[awr] #[ort] #[uer] #[arr])

/ɪ r/ → (#[ear] #[ere] #[eer] #[ier] #[er] #[eare] #[eir])

/z/ → (#[s] #[z] #[es] #[ze] #[se] #[*] #[x] #[zz] #[z_e] #[s_e] #[ss])

/s/ → (#[s] #[ss] #[t] #[se] #[ce] #[c] #[sc] #[*] #[st] #[sse] #[ste] #[z] #[c_e] #[ps] #[s_e])

/š/ → (#[sh] #[ti] #[ss] #[ch] #[t] #[s] #[*] #[sch])

/ž/ → (#[z] #[s] #[ge] #[t] #[ti] #[si] #[g])

/f/ → (#[ph] #[f] #[gh] #[fe] #[ff])

/θ/ → #[th]

/ð/ → (#[th] #[the])

/v/ → (#[v] #[ve] #[v_e] #[ph] #[f])

/p/ → (#[p] #[pe] #[pp] #[p_e])

/t/ → (#[t] #[te] #[ed] #[d] #[tt] #[th] #[dt] #[tte] #[*] #[t_e] #[z] #[s] #[pt])

/k/ → (#[c] #[k] #[ck] #[q] #[ke] #[che] #[x] #[ch] #[que] #[k_e] #[qu] #[cc])

/d/ → (#[d] #[de] #[ed] #[dd] #[ddh] #[d_e] #[z] #[zz])

/b/ → (#[b] #[be] #[bb] #[b_e)

/g/ → (#[g] #[gg] #[x] #[gue] #[gh] #[gu])

/ǰ/ → (#[ge] #[j] #[dge] #[g] #[dg] #[gg] #[g_e])

/č/ → (#[ch] #[tch] #[che] #[t] #[c])

/m/ → (#[m] #[me] #[mn] #[mb] #[mm] #[mp] #[m_e])

/m̩/ → #[m]

/n/ → (#[n] #[ne] #[kn] #[nn] #[nne] #[gn] #[gne] #[on] #[n_e] #[mn])

/n/ → (#[ng] #[n])

PRE-MARKER → =

ROOT-MARKER → ==

ISUF-MARKER → ++

DSUF-MARKER → +

COMPOUND-MARKER → $

SSYL1-MARKER → !

SSYL2-MARKER → ?

M-ONSET-MARKER → @

Appendix H

Nonparsable Words

This appendix lists examples of nonparsable words in letter-to-sound and sound-to-letter generation.

H.1 Nonparsable Words in Letter-to-sound Generation

AESTHETIC
AH
ALBUMIN
ANSWERED
ARCHAEOLOGY
BATHROOM
BOYCOTT
CALCIUM
CARNEGIE
CHAMPAGNE
CHARLIE

COCKTAIL

DAWN

DRIVEWAY

DYLAN

EH

EMBASSY

EVERYDAY

FAIRLY

FIERCE

FOOTBALL

FULFILLMENT

HANDKERCHIEF

HAY

HENRIETTA

JOE

JOYCE

KATIE

KERN

KIRBY

LLOYD

LUGGAGE

MCCLELLAN

MILKMAN

NEWT

OUTGOING

REORGANIZATION

SERIOUSNESS

SETTLING

SHEAR

SHOE

SIOUX

STUBBORN

THIGH

THURSDAY

TIGHTLY

TOUGHER

TYPEWRITER

UPWARD

WARSAW

WHOLLY

ZINC

H.2 Nonparsable Words in Sound-to-letter Generation

AESTHETIC

ALBUMIN

ARCHAEOLOGY

BATHROOM

BEGGED

BOYCOTT

CARNEGIE

COALITION

CONTINUITY

CUSHION

DRAGGED

DRIVEWAY

ENTHUSIASM

EVERYDAY

FAIRLY

FIERCE

FOOTBALL

GIGANTIC

HANDKERCHIEF

HENRIETTA

JANUARY

LABORATORIES

MANAGEMENT

MILKMAN

MIMESIS

NEVERTHELESS

NIGHTMARE

OUTGOING

PENINSULA

PICTURESQUE

PREJUDICE

PROJECT

REORGANIZATION

RESIDUE

ROUTINE SEGREGATION

SERIOUSNESS

SHRUGGED

SMOOTH

THEOREM

TYPEWRITER

UNNECESSARY

UPWARD

VAGINA

WITHOUT

Bibliography

[1] Adams, M., “What Good is Orthographic Redundancy?,” *Perception of Print*, H. Singer and O. Tzend (Eds.), pp. 197-221, Hillsdale, NJ:Erlbaum, 1981.

[2] Allen, J., “Synthesis of speech from unrestricted text,” *Proc. IEEE*, vol. 64, pp. 422-433, 1976.

[3] Allen, J., S. Hunnicutt and D. Klatt, *From Text to Speech: The MITalk System,* Cambridge University Press, Cambridge, 1987.

[4] Alleva, F. and K. F. Lee, “Automatic New Word Acquisition: Spelling from Acoustics,” *Proc. of the DARPA Speech and Natural Language Workshop,* October 1989.

[5] Antworth, E., *PC KIMMO: A Two-Level Processor for Morphological Analysis,* Summer Institute of Linguistics, Inc., 1990.

[6] Asadi, A., *Automatic Detection and Modeling of New Words in a Large-Vocabulary Continuous Speech Recognition System,* Ph.D. thesis, Department of Electrical and Computer Engineering, Northeastern University, Boston, August 1991.

[7] Bernstein, J. and D. Pisoni, “Unlimited Text-to-speech System: Description and Evaluation of a Microprocessor based device,” *Proc. ICASSP-80*, pp. 576-579, Denver, 1984.

[8] Bernstein, J. and D. Rtishchev, "A Voice Interactive Language Instruction System," *Proc. Eurospeech-91*, pp. 981-984, Italy, 1991.

[9] Brill, E., "A simple rule-based part-of-speech tagger," *Proc. of the Third Conference on Applied Natural Language Processing*, Association of Computational Linguistics, Trento, Italy, 1992.

[10] Byrd R. and M. Chodorow, "Using an On-line Dictionary to Find Rhyming words and Pronunciations for Unknown Words," *Proc. ACL*, pp. 277-283, Chicago, 1985.

[11] Chomsky, N. and M. Halle, *Sound Pattern of English*, New York, Harper & Row.

[12] Church, K., "Stress Assignment in Letter to Sound Rules for Speech Synthesis," *Proc. ACL*, pp. 246-253, Chicago, 1985.

[13] Church, K., "Morphological Decomposition and Stress Assignment for Speech Synthesis," *Proc. ACL*, pp. 156-164, New York, 1986.

[14] Church, K., *Phonological Parsing in Speech Recognition,* Kluwer Academic Publishers, 1987.

[15] Cohen, M., G. Baldwin, J. Bernstein, H. Murveit and M. Weintraub, "Studies for an Adaptive Recognition Lexicon," Proc. DARPA Speech Recognition Workshop, Report No. SAIC-87/1644, pp. 49-55, February 1987.

[16] Coker, C., K. Church and M. Liberman, "Morphology and Rhyming: Two Powerful Alternatives to Letter-to-Sound Rules for Speech Synthesis," *Proc. of the Conference on Speech Synthesis*, European Speech Communication Association, 1990.

[17] Conroy, D., T. Vitale and D. Klatt, *DECtalk DTC03 Text-to-Speech System Owner's Manual,* Educational Services of Digital Equitpment Corporation, P.O. Box CS2008, Nashua, NH 03061, 1986. Document number EK-DTC03-OM-001.

[18] Damper, R., "Self-Learning and Connectionist Approaches to Text-Phoneme Conversion," *Proc. of 2nd Neural Computation and Psychology Workshop,* edited by J. Levy, Forthcoming.

[19] Dedina, M. and H. Nusbaum, "PRONOUNCE: A Program for Pronunciation by Analogy," *Computer Speech and Language*, Vol. 5, No. 1, pp. 55-64, January 1991.

[20] Denes, P. and E. Pinson, *The Speech Chain: The Physics and Biology of Spoken Language*, W. H. Freeman and Company.

[21] Elovitz, H., R. Johnson, A. McHugh and J. Shore, *Automatic Translation of English Text to Phonetics by means of Letter-to-Sound Rules,* Naval Research Laboratory Technical Report 7949, January 1976.

[22] Fant, G., "Analysis and Synthesis of Speech Processes," *Manural of Phonetics*, ed. B. Malmberg, Chapter 8, pp. 173-277, North-Holland Publishing Co., 1970.

[23] Flammia, G., J. Glass, M. Phillips, J. Polifroni, S. Seneff and V. Zue, "Porting the Bilingual Voyager System to Italian," *Proc. ICSLP-94*, pp. 911-914, Yokohama, 1994.

[24] Glass, J., D. Goodine, M. Phillips, S. Sakai, S. Seneff and V. Zue, "A Bilingual VOYAGER System," *Proc. Europeech-93*, pp. 2063-2066, September 1993.

[25] Glass, J., J. Polifroni and S. Seneff, "Multilingual Language Generation Across Multiple Domains," *Proc. ICSLP-94*, pp. 983-986, Yokohama, 1994.

[26] Glushko, R., "The organization and activation of orthographic knowledge in reading aloud," *Journal of Experimental Psychology: Human Perception and Performance,* Vol. 5, pp. 674-691, 1979.

[27] Goddeau, D., personal communication.

[28] Goddeau, D., E. Brill, J. Glass, C. Pao, M. Phillips, J. Polifroni, S. Seneff and V. Zue, "Galaxy: A Human-Language Interface to On-line Travel Information," *Proc. ICSLP-94*, pp. 707-710, Yokohama, 1994.

[29] Golding, A., *Pronouncing Names by a Combination of Case-based and Rule-based Reasoning,* Ph.D. Thesis, Stanford University, 1991.

[30] Golding, A. and P. Rosenbloom, "A Comparison of ANAPRON with Seven Other Name-pronunciation Systems," *Journal of the Americal Voice I/O Society,* August 1993, pp.1-21.

[31] Goodine, D. and V. Zue, "Romaine: A Lattice Based Approach to Lexical Access," *Proc. Eurospeech-93*, pp., Berlin, Germany, 1993.

[32] Groner, G., J. Bernstein, E. Ingber, J. Perlman and T. Toal, "A Real-time Text-to-speech Converter," *Speech Technology*, 1, 1982.

[33] Hauptmann, A., J. Mostow, S. Roth, M. Kane, A. Swift, "A Prototype Reading Coach that Listens: Summary of Project LISTEN," *Proc. of the ARPA Human Language Technology Workshop,* pp. 237-238, New Jersey, 1994.

[34] Hertz, S., J. Kadin and K. Karplus, "The Delta rule development system for speech synthesis from text," *Proc. of the IEEE* Vol. 73, No. 11, pp. 1589-1601, 1985.

[35] Hetherington, Lee, *A Characterization of the Problem of New, Out-of-Vocabulary Words in Continuous Speech Recognition and Understanding,* Ph.D. thesis, MIT, August 1994.

[36] Hochberg, J., S. M. Mniszewski, T. Calleja and G. J. Papcun, "A Default Hierarchy for Pronouncing English," *IEEE Transactions on Pattern Matching and Machine Intelligence*, Vol. 13, No. 9, pp. 957-964, September 1991.

[37] Huang, C., M. Son-Bell and D. Baggett, "Generation of Pronunciations from Orthographies using Transformation-based Error-driven Learning," *Proc. ICSLP*, pp. 411-414, Yokohama, Japan, 1994.

[38] Hunnicutt, S., "Phonological Rules for a Text-to-speech System," *American Journal of Computational Linguistics*, AJCL Microfiche 57, 1976.

[39] Jelinek, F. "Up from Trigrams! The Struggle for Improved Language Model," *Proc. Eurospeech-91*, pp.1037-1041, Genova, Italy, 1991.

[40] Klatt, D. and D. Shipman, "Letter-to-Phoneme Rules: A Semi-automatic Discovery Procedure," *J. Acoust. Soc. Am.,* 82, pp.737-793, 1982.

[41] Klatt, D., "Review of Text-to-speech Conversion for English," JASA 82 (3), Acoustic Society of America, pp. 737-793, 1987.

[42] Kompe, R. et al, "Prosody Takes Over: Towards a Prosodically Guided Dialog System," *Speech Communication*, Vol. 15, pp. 153-167, 1994.

[43] Kucera, H. and W. N. Francis, *Computational Analysis of Present-Day America English*, Brown University Press, 1967.

[44] Lamel, L. F., R. H. Kassel and S. Seneff, "Speech Database Development: Design and Analysis of the Acoustic-Phonetic Corpus," *Proc. DARPA Speech Recognition Workshop*, Report No. SAIC-86/1546, pp. 100-109, February 1986.

[45] Lee, K. F., *Automatic Speech Recognition: The development of the SPHINX system,* Kluwer Publishers, 1989.

[46] Lehnert, W., "Case-based problem solving with a Large Knowledge Based of Learned Cases," *Proc. AAAI-87*, pp. 301-306, Seattle, 1987.

[47] Lesser, V., R. Fennell, L. Erman and R. Reddy, "The Hearsay II Speech Understanding System," *IEEE Transactions on Acoustics Speech and Signal Processing*, ASSP-23(1), pp. 11-24, February 1975.

[48] Lesser, V., F. Hayes-Roth, M. Birnbaum and R. Cronk, "Selection of Word Islands in the Hearsay-II Speech Understanding System," *Proc. ICASSP-77*, pp. 791-794, Hartford, 1977.

[49] Lowerre, B. and R. Reddy, "The Harpy Speech Understanding System," *Trends in Speech Recognition,* Prentice-Hall, Englewood Cliffs, NJ, 1980.

[50] Lucas, S. and R. Damper, "Syntactic Neural Networks for Bi-directional Text-phonetics Translation," in *Talking Machines, theories, models and designs,* G. Bailly and C. Benoit (Eds.), pp. 127-142, North-Holland Publishers, 1992.

[51] Lucassen, J. and R. Mercer, "An Information Theoretic Approach to the Automatic Determination of Phonemic Baseforms," *Proc. ICASSP-84*, pp. 42.5.1-42.5.3, San Diego, 1984.

[52] Luk, R. and R. Damper, "Inference of Letter-Phoneme Correspondences with Pre-defined Consonant and Vowel Patterns," Proc. ICASSP-93, pp. 203-206, Minneapolis, 1993.

[53] McClelland, J. L. and M. S. Seidenberg, "A Distributed Developmental Model of Word Recognition and Naming," *Psychological Review*, Vol. 96, No. 4, pp. 523-568, 1989.

[54] McCandless, M., *Word Rejection for a Literacy Tutor,* S. B. Thesis, M.I.T., 1992.

[55] McCulloch, N., M. Bedworth and J. Bridle, "NETspeak — a reimplementation of NETtalk," *Computer Speech and Language*, Vol. 2, pp. 289-301, 1987.

[56] Meng, H., S. Seneff and V. Zue, "Phonological Parsing for Bi-directional Letter-to-Sound/Sound-to-Letter Generation," *Proc. ARPA HLT-94*, New Jersey, 1994.

[57] Meng, H., S. Seneff and V. Zue, "The Use of Higher Level Linguistic Knowledge for Spelling-to-Pronunciation Generation," *Proc. ISSIPNN-94*, pp. 670-673, Hong Kong, 1994.

[58] Meng, H., S. Seneff and V. Zue, "Phonological Parsing for Reversible Letter-to-Sound/Sound-to-Letter Generation," *Proc. ICASSP-94*, pp. II-1 to II-4, Adelaide, Australia, 1994.

[59] Meng, H. M., S. Hunnicutt, S. Seneff, V. W. Zue, "Phonological Parsing for Bi-directional Letter-to-Sound / Sound-to-Letter Generation," submitted to the *Journal of Speech Communication*, October 1994.

[60] Nilsson, J., *Problem-solving methods in artificial intelligence,* Computer Science Series, McGraw-Hill.

[61] Oakey, S. and R. Cawthorne, "Inductive Learning of Pronunciation Rules by Hypothesis Testing and Correction," *Proc. IJCAI-81*, pp. 109-114, Vancouver, Canada, 1981.

[62] Oshika, B., V. Zue, R. Weeks, H. Nue and J. Auerbach, "The Role of Phonological Rules in Speech Understanding Research," *IEEE Transactions on ASSP*, Vol. ASSP-23, pp. 104-112, 1975.

[63] Pallet, D., J. Fiscus, W. Fisher, J. Garofolo, B. Lund and M. Przbocki, "1993 Benchmark Tests for the ARPA Spoken Language Program," *Proc. of the ARPA Workshop on Human Language Technologyc*, pp. 49-54, March 1994.

[64] Parfitt, S. and R. Sharman, "A Bi-directional Model of English Pronuciation," *Proc. Eurospeech*, pp. 801-804, 1991.

[65] Pierraccini, R., Z. Gorelov, E. Levin and E. Tzoukermann, "Automatic Learning in Spoken Language Understanding," *Proc. ICSLP-92*, pp. 405-408, Banff, Canada, 1992.

[66] Price, P., "Evaluation of Spoken Language Systems: the ATIS Domain," *Proc. of the DARPA Speech and Natural Language Workshop,* pp. 91-95, Hidden Valley, Pennsylvania, 1990.

[67] Rabiner, L., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Readings in Speech Recognition,* A. Waibel and K. F. Lee (Eds.), Morgan Kaufman Publishers, 1990.

[68] Rosson, M., "The Interaction of Pronunciation Rules and Lexical Representation in Reading Aloud," *Memory and Cognition,* Vol. 13, pp. 90-99, 1985.

[69] Rozin, P. and L. Gleitman, "The structure and acquisition of reading II: The reading process and the acquisition of the alphabetic principle," *Towards a psychology of reading,* A. Reber & D. Scarborough (Eds.), (pp. 55-141). Hillsdale, NJ:Erlbaum, 1977.

[70] Russell, N. H., personal communication.

[71] Segre, A., Sherwood, B. and W. Dickerson, "An Expert System for the Production of Phoneme Strings from Unmarked English Text using Machine-induced Rules," *Proc. First European ACL*, Pisa, Italy, pp. 35-42, 1983.

[72] Sejnowski, T. J. and C. R. Rosenberg, "NETtalk: parallel networks that learn to pronounce English text," *Complex Systems,* 1, pp. 145-168, 1987.

[73] Seneff, S., "TINA: A Natural Language System for Spoken Language Applications," *Computational Linguistics,* Vol. 18, No. 1, pp. 61-86, March 1992.

[74] Seneff, S., H. Meng and V.Zue, "Language Modeling using Layered Bigrams," *Proc. ICSLP-92*, pp. 317-320, Banff, Calgary, Canada.

[75] Shipman, D. and V. W. Zue, "Properties of large lexicons: Implications for advanced isolated word recognition systems," *Proc. ICASSP-82*, pp. 546-549, 1982.

[76] Silverman, K., E. Blaauw, J. Spitz and J. Pitrelli, "A Prosodic Comparison of Spontaneous Speech and Read Speech," *Proc. ICSLP-92*, pp. 1299-1302, Banff, Canada, 1992.

[77] Spiegel, M. and M. Macchi, "Synthesis of Names by a Demisyllable-based Speech Synthesizer (Orator)," *Journal of the American Voice Input/Output Society,* 7, 1990. Special RHC/RBOC issue.

[78] Stanfill, C. and D. Waltz, "Toward Memory-Based Reasoning," *Communications of the ACM,* 12(12), pp. 1213-1228, December 1986.

[79] Stanfill, C., "Memory-Based Reasoning Applied to English Pronunciation," *Proc. AAAI-87*, pp. 577-581, 1987.

[80] Stanfill, C., "Learning to Read: A Memory-based Model," *Proc. of the Case-based Reasoning Workshop*, Clearwater Beach, FL, pp. 406-413, 1988.

[81] Stevens, K., Unpublished course notes for *Speech Communication*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Spring term, 1989.

[82] B. Suhm, M. Woszczyna, and A. Waibel, "Detection and transcription of new words," in *Proc. European Conf. Speech Communication and Technology*, pp. 2179–2182, Berlin, September 1993.

[83] Sullivan, K. and R. Damper, "Novel-word Pronunciation Within a Text-to-speech System," in *Talking Machines, theories, models and designs*, G. Bailly and C. Benoit (Eds.), pp. 183-195, North Holland Publishers, 1992.

[84] *The Carnegie Mellon Pronouncing Dictionary* [cmu-dict.0.1], R. L. Weide and P. Jansen (Eds.), Carnegie Mellon University, Pittsburgh, USA, 1993.

[85] van Coile, B. M. J. "The Depes development system for text-to-speech synthesis," *Proc. ICASSP-89*, pp. 250-253, Glassgow, Scotland, May 1989.

[86] van Coile, B. "Inductive Learning of Grapheme-to-Phoneme rules," *Proc. ICSLP-90* pp. 765-768, Kobe, Japan, 1990.

[87] van Coile, B., S. Lyes and L. Mortier, “On the Development of a Name Pronunciation System,” *Proc. ICSLP-92*, pp. 487-490, Banff, Canada, 1992.

[88] van den Bosch, A. and W. Daelemans, ”Data-Oriented Methods for Grapheme-to-Phoneme Conversion,” *Proc. Sixth European ACL*, pp. 45-53, 1993.

[89] van Leeuwen, H. C., “Speech Maker Formalism: a rule formalism operating on a multi-level, synchronized data structure,” *Computer Speech and Language,* Volume 7, Number 4, October 1993.

[90] Vitale, T., “Foreign Language Speech Synthesis: Linguistics and Speech Technology,” *Proc. of the Voice I/O System Applications Conference*, pp. 363-370, San Francisco, 1985.

[91] Wang, W. J., W. N. Campbell, N. Iwahashi & Y Sagisaka, “Tree-based Unit Selection for English Speech Synthesis,” *Proc. ICASSP-93*, pp. II-191 to II-194.

[92] Ward W., and S. Issar, “Recent Improvements in the CMU Spoken Language Understanding System,” *Proc. of the ARPA Workshop on Human Language Technology*, pp. 213-216, March 1994.

[93] Weintraub, M., and J. Bernstein, “RULE: A System for Constructing Recognition Lexicons,” Proc. DARPA Speech Recognition Workshop, Report No. SAIC-87/1644, pp. 44-48, February 1987.

[94] Woszczyna, M. *et al,* “Towards Spontaneous Speech Translation,” *Proc. ICASSP-94*, pp. 345-348, Adelaide, Australia, 1994.

[95] Yannakoudakis, E. and P. Hutton, “Generation of spelling rules from phonemes and their implications for large dictionary speech recognition,” *Speech Communication*, Vol. 10, pp. 381-394, 1991.

[96] Zue, V., “The Use of Speech Knowledge in Automatic Speech Recognition,” *IEEE Proceedings*, Vol. 73, No. 11, pp. 1062-1615, November 1985.

[97] Zue, V., J. Glass, M. Phillips and S. Seneff, "The MIT SUMMIT Speech Recognition System: a Progress Report," *Proc. ARPA Speech and Natural Language Workshop*, pp. 21-23, Philadelphia, 1989.

[98] Zue, V., J. Glass, D. Goodine, H. Leung, M. Phillips and S. Seneff, "The VOYAGER Speech Understanding system: Preliminary Development and Evaluation," in *Proc. ICASSP-90*, pp. 73-76, Albuquerque, NM, April 1990.

[99] Zue, V., J. Glass, D. Goodine, M. Phillips and S. Seneff, "The SUMMIT Speech Recognition System: Phonological Modelling and Lexical Access," *Proc. ICASSP*, pp. 49-52, Albuquerque, NM, 1990.

[100] Zue, V., J. Glass, D. Goddeau, D. Goodine, C. Pao, M. Phillips, J. Polifroni and S. Seneff, "PEGASUS: A Spoken Dialogue Interface for On-line Air Travel Planning," *Speech Communication*, Vol. 15, pp. 331-340, 1995.