Serial Control of Phonology in Speech Production

Janet Vousden

Submitted for the qualification of Degree of PhD at The University of Warwick, Department of Psychology, February 1996.

Table of Contents

Table of Figures

Table of Tables

Acknowledgements

Firstly, I would like to thank Trevor Harley for providing sound and consistent advice and an excellent source of knowledge throughout my time as a postgraduate in the Department whilst working on my thesis. I am indebted to him for allowing me full access to his speech error corpus on which I based my analysis of speech errors. I would also like to acknowledge his commitment to ensuring more than adequate access to resources and his efficienct handling of administrative matters. I would also like to thank Gordon Brown firstly for agreeing to act as my second supervisor sometime after I commenced this research, and secondly for providing hours of stimulating discussion, without which many ideas would surely have laid dormant. I am indeed very grateful for many an hour spent listening to my thoughts and for providing guidance on relevant material. Thanks must also go to the Economic and Social Research Council for their financial support.

I wish to thank Greg Milligan for his excellent technical support of the computing facilities during his time at Warwick. I would like to thank Siobhan MacAndrew, Arlene Astelle, Helen Bown, and especially Sarah Norgate for their moral support, understanding and keeping me inspired and motivated.

Far from least, my mum, dad, Graeme, and all my friends deserve many thanks for their support and understanding of how busy and consumed I have been in completing this thesis. Finally I would like to thank Dave, to whom I have dedicated this thesis, for without his extraordinary enthusiasm I am sure I would never have began. My words of thanks to him seem quite inadequate for the continued support and inspiration he has given me over the years: both day and night, through good times and bad.

Declaration

I declare that this thesis is my own work. No part of it has been submitted for either qualification at another institution or as a publication to a journal and neither has any of it been presented at any conference or appeared in any conference proceedings.

To Dave

Summary

The aim of this thesis is to further our understanding of the processes which control the sequencing of phonemes as we speak: this is an example of what is commonly known as the serial order problem. Such a process is apparent in normal speech and also from the existence of a class of speech errors known as sound movement errors, where sounds are anticipated (spoken too soon), perseverated (repeated again later), or exchanged (the sounds are transposed). I argue that this process is temporally governed, that is, the serial ordering mechanism is restricted to processing sounds that are close together in time. This is in conflict with frame-based accounts (e.g. Dell, 1986; Lapointe & Dell, 1979), serial buffer accounts (Shattuck-Hufnagel, 1979) and associative chaining theories (Wickelgren, 1969).

An analysis of sound movement errors from Harley and MacAndrew's (1995) corpus shows how temporal processing bears on the production of speech sounds by the temporal constraint observed in the pattern of errors, and I suggest an appropriate computational model of this process. Specifically, I show how parallel temporal processing in an oscillator-based model can account for the movement of sounds in speech. Similar predictions were made by the model to the pattern of movement errors actually observed in speech error corpora. This has been demonstrated without recourse to an assumption of frame and slot structures. The OSCillator-based Associative Recall (OSCAR) model, on the other hand, is able to account for these effects and other positional effects, providing support for a temporal based theory of serial control.

Chapter 1

1. Introduction

This thesis aims to further the understanding of the speech production system by investigating the problem of the serial ordering of phonemes within the framework of connectionism. The act of speech can be viewed crudely as the process of translating an idea into the appropriate sequence of speech sounds. This is undoubtedly a complex task, involving more than one process. I focus upon the serial control of a phonological representation given the successful access of lexical items. How does the process of speaking begin? How are words chosen, ordered and finally translated into a sequence of appropriate phonemes? How is it that the sequence of phonemes is controlled during production so that they are spoken in the correct order?

1.1 Aims

This thesis primarily focuses on the final question: how is it that speech sounds are spoken as a sequence of correctly ordered items? Words may exist as single entities but when they are strung together to form a spoken message they must be broken down into a temporal sequence of speech-motor actions. Apart from speech error data providing a clear indication of a serial ordering mechanism, the normal process of retrieving the phonetic form of words from the mental lexicon strongly suggests that there must exist a mechanism for serial order. For example, the three words "cat", "act", tack", are realised by producing the same phonemes in different orders. It is well within our capability to produce the correct order of sounds for each

different word. A mechanism for serial order is clearly implicated as a fundamental part of the speech production system and this thesis aims to investigate the basic form and possible principles underlying such a mechanism. Studies of spontaneously occurring speech errors, or slips of the tongue, also indicate that a mechanism for producing serial order must exist as part of the production system. This is because when errors occur, the order of words or sounds can be disrupted and the disruption to the ordering follows systematic patterns. For words to be understood each sound must be produced in the correct order. When speech errors occur, sounds may be misordered, deleted or added, resulting in a sequence of sounds that deviates from the desired utterance. Serial order is not preserved in these cases and they are classified as an error of sound ordering somewhere within the speech production system. Specifically, the existence of a class of sound errors characterised as *contextual* or *movement* errors suggests the existence of a mechanism that controls the output order of sounds during production. This process is not infallible and occasionally it malfunctions to produce such errors. Sound movement errors are broadly classified into three types: *anticipations*, *perseverations* and *exchange* errors. A sound is *anticipated* if it is produced prematurely, as in [1.1]. A sound is *perseverated* if it is repeated accidentally at a later stage, as in [1.2]. Finally, sounds *exchange* if they effectively swap places during production, as in [1.3]. Errors [1.1] to [1.3] are from Harley and MacAndrew (1995).

[1.1] autowatic washing machine -> automatic washing machine

[1.2] is it fish and **ch**ip **ch**ime yet? -> is it fish and chip time yet?

[1.3] **p**ick as a sarrot -> sick as a parrot

The theoretical basis of a serial order mechanism is investigated using an implementational (computational) approach. This thesis presents a series of

connectionist models of serial order with the aim of providing an account of the serial order error data apparent in the production of speech.

1.2 Speech Production: A Summary

Research into speech recognition has largely outweighed research into the production of speech. This may be due to the fact that it is easier to control the input of experimental language recognition tasks than it is to control it in production tasks. To characterise the process(es) of speaking, one must control the thoughts (i.e. the input) that are ultimately translated into an ordered sequence of sounds (i.e. the output). This is obviously more difficult than providing input for a speech recognition task, where the subject actually hears the spoken word. In studies of speech recognition, the experimenter is in control of the words the subject hears, and the input is real and easily described. Serial order is not such an important issue in speech recognition because order is explicitly stated in the auditory signal (i.e. the spoken word(s)). In production, the input (i.e. thoughts) is more abstract and less easily described, to say the least.

The additional task of imposing order on the output further complicates the process of speaking because the correct information not only must be recalled, it must also be appropriately ordered. Suggested mechanisms specifically for the ordering of speech sounds have often been avoided by making explicit reference to pre-conceived devices or representations within the architecture. For example, many theories (e.g. Lapointe & Dell, 1989) assume the existence of *phonological frames* which provide empty slots to be filled with specific types of information. Hence the order in which the phonemes are articulated is provided by the frame itself. For example, a phonological frame may specify the shape of the word, the number of phonemes and the type of information (e.g. a consonant or vowel) that may be inserted into each of the slots in the frame. Other theories (e.g. Shattuck-Hufnagel, 1979) assume an

output buffer which incrementally stores a sequence until it is ready for output, allowing editing of the speech signal before it is articulated to remove illegal or suspect utterances. These devices that avoid the issue of how temporal order is achieved will be covered in greater detail in subsequent chapters. Many of these mechanisms do not directly address the problem of how serial order is derived in the first place, as it is merely implied by the infrastructure (e.g. a phonological frame) in the models. Thus serial order is produced as a consequence of representational assumptions, without any specification of the processes that give rise to them or their temporal nature.

Serial ordering mechanisms are not limited to the ordering of sounds alone. This can be seen by the different types of speech errors that are often made. Speech can be broken down into different levels of units, from whole sentences down through words and syllables to individual phonemes and sounds. Clearly, misordering of any of these units will produce errors, but different types of errors.

[1.4] The **eye**'s shining on your **sun.**

For example, an incorrect ordering of words, such as in [1.4] is quite different from an incorrect ordering of phonemes within a word, such as the examples given in [1.1] - [1.3]. These different types of speech error suggest that serial ordering occurs at different *levels* on different units within the production system, although it is by no means clear that the same mechanism or process is responsible for the serial ordering at the various levels. For instance, the correct sequencing of words within sentences is determined by the syntax of the language, although syntax has little effect upon the correct phoneme sequence that makes up a phonological sequence of speech sounds. Partly for this reason, the scope of this thesis is limited to sound movement errors rather than larger units of speech. Garrett (1975) conducted extensive research into the notion of levels in sentence production, using the pattern of speech errors known as word exchange errors, versus spoonerisms, or sound exchange errors (as in [1.3])

as the basis for his theory that there exist different levels of representation. He also proposed that processing occurred in separate stages, which were confined to the representations found at each level. Each level is said to give rise to certain types of speech error and also to constrain the way in which they can happen. A significant finding was that there was a difference in the distance different error types span. Smaller units of speech (phonemes) were constrained by distance, and larger units (words) could move much greater distances, but instead were constrained by grammatical class. This observation is particularly relevant to this thesis because it is an important factor for a serial ordering mechanism. Speech is also constrained by other factors, such as phonotactic, structural and form class properties. I will return to discuss these other constraints in detail later. The focus of this thesis is on the serial ordering of phonemes during production, and how an ordering mechanism could account for the constraint on the distance over which sound movement errors occur. It is motivated by the distribution of sound movement speech errors displaying serial order effects in Harley and MacAndrew's (1995) corpus.

1.3 Theoretical Framework

The pattern of movement errors in Harley and MacAndrew's (1995) corpus is consistent with the patterns reported from other corpora. The distance constraint on exchanges is consistent with Garrett (1975) and MacKay's (1970) analysis of other corpora but for Harley and MacAndrew's (1995) corpus is also found to exist for anticipation and perseveration errors. An implementational explanation of serial order effects was favoured over a theoretical account as an implementation is more readily testable. The method of implementation follows a connectionist approach. Connectionism provides a good framework to implement and test theories of processing and it is for this reason that it has been chosen to model a serial order mechanism for speech sounds. Connectionism also possesses many appealing

qualities which make it a desirable framework within which to investigate speech production. For instance, complex non-linear problems can be solved through multiple constraint satisfaction, solutions are found by exposure to example data rather than explicit rule application, statistical properties of the data can be discovered and usefully incorporated as part of the knowledge of the system, and connectionism also allows for noisy or damaged systems to provide an appropriately degraded response, rather than a complete blank. Good connectionist theories also provide testable predictions. For example, Dell's (1989) model predicted that repeated phonemes in monosyllabic word pairs would increase the probability of producing non-target phonemes in both adjacent *and* non-adjacent positions to the repeated phoneme. This prediction was subsequently borne out experimentally, providing support for the model.

Alternative choices of methodology include performing controlled experiments on human subjects under specific conditions. For example, Dell (1989) used an experimental paradigm which was designed to elicit speech errors in controlled environments (although Baars, Motley and MacKay (1975) were the first to use such a technique). This approach deals directly with human subjects, although it is still hard to be certain of the thoughts or processes that precipitate speech errors in real people even under experimental control.

The general architecture used for the model(s) in this thesis is psycholinguistically motivated, assuming that the input to the model(s) is in an abstract form embodying the same information as an abstract lexical representation, or lemma, in Harley's (1990) model. The output is phonetically represented in the form of speech sounds. There have been several different approaches to producing serial order within the connectionist paradigm and this is reflected in the particular implementation of the models presented in the thesis. The models in the early chapters are influenced by Jordan's (1986a, 1986b) recurrent network approach to

serial order, and the models presented in later chapters are more in the style of Houghton's (1990) model.

The main body of the thesis consists of a series of computer simulations of connectionist models of serial order in speech production. The thesis aims to describe how a connectionist model can provide an account of the serial order errors found in speech production and subsequently provide an outline of the processes required to order speech sounds.

1.4 Outline

The thesis begins with a review of study methods and existing modelling paradigms. The review is broadly split into two sections and forms the body of chapters 2 and 3. Chapter 2 reviews the different types of research strategies that have been used as a method of investigating speech production in general, focusing specifically on the current literature on analyses of speech errors and the patterns they yield. I present a new analysis of sound movement errors in Harley and MacAndrew's (1995) speech error corpus. Some observed constraints on exchange errors reported by other researchers (e.g. Boomer & Laver, 1968; Garrett, 1975; MacKay 1970) are also found in Harley and MacAndrew's (1995) exchange errors and also in their anticipation and perseveration errors. The most striking observation for a temporal theory of serial order is characterised by the constraint which restricts the distance over which the errors in question are manifest.

In chapter 3, I review theoretical models of speech production, using Garrett's (1975) model as an example of a model of sentence production. The emphasis is on phonological encoding in these models, and more specifically how well they can account for the speech error data analysed in chapter 2. As will be seen, many of these models rely on a phonological frame-based approach to account for serial order.

The idea of a phonological frame approach to account for certain constraints on error data is discussed and found to be wanting.

Chapter 4 reviews some specific modelling techniques that have been used to model serial order, including the technical issues relevant to modelling serial order within connectionism. The chapter concludes with a review of the application of this approach to speech production models, observing that many models still assume a phonological-frame as part of their infrastructure and are therefore not feasible models of serial order. Hence they are the motivation for a model that provides a true mechanism for serial order without recourse to the assumption of a phonological frame as part of the infrastructure.

The first of the connectionist models implemented in this thesis is presented in chapter 5. The first set is of a basic *recurrent network* model of serial order based on that of Jordan (1986a, 1986b). The model learns to represent one item at a time in a sequence on discrete cycles and is applied to a simple sequence task as a demonstration of the process by which it learns serial order. A computational explanation of this process is given and its limitations are discussed in detail. Reasons to develop an improved model are suggested in light of the limitations found.

The new model, in chapter 6, is based on the Adaptively Parameterised Error Correcting System (henceforth APECS) learning algorithm (McLaren, 1993). Chapter 5 contains simulations which test the learning ability of the new model and show it can successfully learn the task in a more computationally efficient way. The model is extended to accommodate representations of information (i.e. words) in a form more akin to the representations likely used in human speech. Further extensions of the model to accommodate these theoretical changes are implemented and discussed. This chapter also contains simulations of single word (speech) production. Although the model is successful in recalling learned words, it possesses various shortcomings which raise doubts about its validity as an account of speech

production. These limitations are discussed at the end of the chapter, which provide further motivations for an alternative implementation in a different vein. The theoretical basis of a serial order mechanism in speech is reviewed in light of these limitations and the model based on the recurrent network paradigm taken in the previous two chapters is abandoned.

Chapter 7 investigates an alternative connectionist topology, based on that of Houghton (1990), which provides a mechanism to order speech sounds in a sequential manner. It also has the property that nearby items are simultaneously represented even though they may not be the current item to be output. The new model, **OSC**illator-based **A**ssociative **R**ecall (henceforth OSCAR) is introduced with some simple simulations of phoneme sequences that demonstrate the temporal sequencing properties of the model. The model is discussed in depth and the serial order mechanism explained. Some limitations of the basic OSCAR model are also discussed, and these motivate modification of the model in the following chapter.

Chapter 8 describes a more realistic model of OSCAR, that aims to account mainly for the constraint that restricts sound errors to nearby phonemes during production. The results of the simulations of this model are discussed with reference to the speech error data. The model is also discussed as an account of the mechanism that gives rise to serial order in speech production. The chapter concludes with a summary of the data that can be accounted for by OSCAR and the limitations of the model as it stands.

Chapter 9 summarises the conclusions drawn from each chapter. The theoretical implications of each implemented model in the thesis are discussed. The limitations of the thesis and opportunities for future research bring the thesis to its conclusion.

Chapter 2

2. Serial Order and Speech Errors

Although speech production is a process often taken for granted by most adult speakers, it is a very complex process. Oral communication requires the production of a correctly ordered sequence of sounds in order for the desired meaning to be correctly perceived by the recipient. The serial ordering of sounds is a fundamental characteristic of coherent speech. Therefore it is not just the problem of choosing how to say something, or which words to use that allows humans to communicate by speaking, but also the skill of producing the correct stream of sounds, one at a time, in a particular order. This is known as the *serial order* problem.

This chapter starts with a brief overview of the approaches to researching speech production in general before reviewing in detail the data that bear on the serial order problem. The review starts with research on the way in which speech is considered to be planned in cognitive cycles (Petrie, 1987). Also mentioned are theories that have developed from experimental response time tasks (Meyer, 1990, 1991), and from neuropsychological evidence (e.g. Kohn & Smith, 1995). Next, I discuss the types of error that arise when the intended order of speech sounds is altered, the apparent pattern of errors they form and the constraints under which they occur (e.g. MacKay, 1970). I then concentrate on the specific pattern of sound movement errors found in Harley and MacAndrew's (1995) error corpus. Finally, the pattern and distribution of movement errors found in this corpus is presented.

2.1 Research Methods in Speech Production

The act of speech can be viewed as the translation of semantic information into a serially ordered set of articulatory commands. This complex task undoubtedly involves more than one process. How does the process of speaking begin? Where do the words come from? What determines their order? How are words translated into a sequence of appropriate phonemes? The organisational possibilities and processing requirements for such a system are numerous. The final product of the system, speech, is easily identified, but the underlying mechanisms at work are somewhat harder to characterise.

The diversity of the skills and knowledge required to speak have attracted research into speech production from more than one discipline, and has been particularly influenced by two. The nature of speech sounds and their development into a structured form has prompted much input from the domain of linguistics (e.g. Chomsky & Halle, 1968). Equal interest has come from psychology in the mental processing and use of speech in everyday life. Thus research has become interdisciplinary in nature and models of speech production are characterised by the integration of more than one discipline. The methodology and evidence provided for the support of various hypotheses differs widely, depending upon the particular aspect of speech production the theory wishes to address, and the tradition within which it is studied. Many sources of data have been drawn from which integrate ideas from both traditions to further the understanding of speech. Petrie (1987) categorised the research methods of speech production into four different approaches: the study of hesitations and pauses, the categorisation and study of speech errors observed in spontaneous speech, experimental control of speech and the study of the effects of brain damage on speaking. The first three approaches have been pursued using both

observational and experimental methods. The last approach is studied within the broader area of cognitive neuropsychology.

2.1.1 Hesitation Analysis

Human speech is for the most part fluent, although there are occasions when a speaker pauses, or hesitates during speech. These breaks in otherwise fluent speech can be informative as to how speech is pre-planned by speakers if the pattern they follow is analysed. Hesitation analysis has been used to study an abstract level of speech production, i.e. a higher level of cognitive processing. It has been presented as evidence for planning and execution phases in speech forming cognitive cycles (Goldman-Eisler, 1958, 1968). The distribution and nature of pauses in natural speech were analysed in terms of their length, frequency and whether they were silent or filled (e.g. with er..., um..., etc.). The different characteristics of pauses suggests they occur for different reasons. For example, filled pauses in speech maybe used to plan the next segment of fluent speech and unfilled pauses, or a hesitant phase, followed by an execution phase (fluent speech) constitutes some sort of cognitive cycle. This particular approach is limited in what it can say about the processes involved in speaking. It does nothing to provide an explicit explanation of the intermediary processes involved in speaking, only claiming that there exist stages in speech where planning takes place and discusses how they relate to the structure of the intended message. Several researchers have nonetheless conducted extensive studies on hesitations and pauses in speech (Petrie, 1987), all related to the higher or more abstract level of planning. Hesitation analysis is not without its empirical limitations. It has been suggested that pauses in speech allow time for listeners to comprehend what is being said. Pauses may well have this purpose, although it is difficult to assess whether this is the sole purpose of particular pauses, or whether the speaker takes advantage of these pauses for additional planning as well.

Hesitations are influenced by other factors, such as the relative difficulty in retrieving a low frequency abstract word, or nervousness in a stressful situation. The tip-of-the-tongue (TOT) state (Brown & McNeill, 1966) that sometimes occurs when a person cannot articulate a particular word is an example of the transient difficulty in recalling the word. In such a state, the speaker is often able to recall certain information about the word, such as the number of syllables, the initial phoneme and so on. Brown and McNeill (1966) were able to induce TOT states in subjects by presenting them with definitions of low frequency content words. This shows that speakers pause for reasons other than cognitive planning of the next piece of speech, such as when they experience transient phonological difficulties in retrieving (already planned) less familiar words (as in TOT's). The analysis of hesitations and pauses has little to say in terms of the processing that ultimately takes place in speech production, but it does make plausible predictions concerning the existence and size of cognitive cycles used in the planning of speech. The size of the planning unit in speech is of importance because it determines what information must be represented and the likely form of that information. The span of speech that is planned at any one time must therefore constrain the way in which parts of the plan can interact or become muddled, as when errors occur.

2.1.2 Experimental Evidence

Other researchers (Baars et al., 1975; Dell, 1989) have used experimental techniques to induce speech errors in controlled situations under different conditions which are then subject to analysis. When phonemes are repeated within words or utterances, the chance of errors occurring is increased, especially for phonemes next to the repeated one. Dell (1989) showed that this repeated phoneme effect is not limited to adjacent phonemes. Phonemes other than neighbours of the repeated phoneme also become more likely to be involved in an error. Also in the

experimental field, Meyer (1990, 1991) used tests of people's ability to respond to paired word cues to provide evidence for the time course of phonological encoding both within and between syllables. Subjects were primed with a word cue, and asked to produce the appropriate paired response word as quickly as possible. Their response times were recorded. When the paired response word shared the same beginning (either syllable onset or entire syllable) as the cue, the response times were significantly faster. The effect was not found when response words shared word final portions with the cues. The results of her experiments supported her hypothesis that syllables are encoded in strict sequential order of their appearance (i.e. from left to right). Hence if subjects were primed with fragments of speech in order, from the left, then they were able to use this information to produce a faster response. If they were primed with fragments not in sequential order (i.e. syllable-final or word-final sounds), then they were not able to use this information because the initial sounds still had to be retrieved first. Therefore, only by priming with initial sounds could their responses be decreased.

2.1.3 Neuropsychological Evidence

Another method of investigation comes from the field of neuropsychology. Kohn and Smith (1994) distinguished between the properties associated with two types of phonological output deficits in aphasics. The first, an activation-deficit, was associated with trouble in activating the stored lexical-phonological representation of words, which refers to difficulty in providing the phonological information stored for a particular lexical entry. The second, a planning-deficit, (which is post-lexical), involved an impairment to the process that transformed the underspecified form from the lexicon into a complete phonemic representation. Kohn and Smith (1995) then investigated the process of phonemic planning of words by comparing the speech errors of six aphasics, three of whom had a planning-deficit and three of whom had an

activation-deficit. Their hypothesis was compatible with Meyer (1990, 1991) to the extent that it views phonemic planning as a sequential process, proceeding from left to right in the order of appearance. They found that all the aphasics produced word initial fragments of speech, but that the aphasics with planning-deficits made consistently more errors in a left to right direction, both from syllable onset through to coda and from one syllable to the next. Thus they found strong serial effects of syllable structure to support their hypothesis that phonemic planning occurs from left to right in serial rather than in parallel. There is no doubt that the execution stage must indeed be strictly sequential, in a left-to-right fashion. The question is at what stage does the execution plan become linearised, and how?

2.1.4 Speech Error Analysis

Spontaneous speech errors or "slips of the tongue" are of interest to the production process, not only for their mere occurrence, but also by providing some insight into the mechanisms at work during normal production. Analyses of error corpora have provided supporting evidence for many important theoretical issues in the processes involved in speech production (e.g. Dell, 1986; Dell & Reich, 1981; Garrett, 1975, 1976; Harley, 1984; Shattuck-Hufnagel, 1979; Shattuck-Hufnagel & Klatt, 1979; Stemberger, 1982, 1985).

However, analyses based on error corpora are not without their problems, as has been well documented in the literature (Cutler, 1982; Ferber, 1990; Meyer, 1992). The main problem with spontaneous error collection concerns the accuracy of the collection. Not all errors are noticed by the collector in the first place (the *detectability problem*) and even the noted errors are subject to erroneous recording, either because they were incorrectly recalled by the collector or incorrectly heard. Cutler (1982) focused on the problems of error detectability and potentially confounding factors which can effect them. Research from the domains of hearing

errors, shadowing and mispronunciation detection, perceptual confusion and the salience of different elements of word structure reveal how real the problem of detectability is. The existence of hearing errors (e.g. Browman, 1980) shows how actual speech errors can easily be perceived (or misperceived) differently by different people. Another problem arises because detectability interacts with error types. The relative position of the error (for example with respect to the syllable structure) and the size of the unit involved in the error are two error characteristics that have different detection rates. For example, sound errors in syllable-initial position or in stressed syllables are noticed more easily whereas errors involving stress and intonation are less often noted (Meyer, 1992). Empirical work by Miller and Nicely (1955) have shown that place of articulation is the least stable articulatory dimension in speech perception. When forced to guess at the identity of consonants masked with noise, subjects were more likely to guess at a consonant that differed only in the place of articulation rather than one that differed along another dimension. These results predict that consonant errors that differ in place of articulation only would be the hardest to detect. However, the converse has been observed in the error data in that other articulatory dimensions (voicing, nasality, manner of articulation) are more likely to be preserved than place of articulation (Shattuck-Hufnagel & Klatt, 1979) and a greater number of sound errors differ only in place of articulation. This could suggest that these errors are not particularly susceptible to perceptual confusion. On the other hand it could also be the case that perceptual confusion does have an effect and many more of these errors go unperceived such that their real frequency is actually under-represented in error corpora. Either way it draws attention to the caution which must be taken in basing hypothesis on the relative proportions of error types. Another problem with relative frequencies of types of error arise from classification strategies for ambiguous errors which may blur the distribution of error types.

However, validation of error corpora can be achieved, or maximised. Corpora obtained from recorded material and the use of large corpora minimises the significance of erroneously noted data. The London-Lund corpus (Garnham, Shillcock, Brown, Mill & Cutler, 1980) is one such collection obtained from recorded material. It has the advantage over other corpora that it can be used to provide an estimate of frequency of certain error types in normal speech. It is also consistent with the types of errors found in other corpora, going some way to validate the reliability of other collections. All corpora largely give the same results, which is also supporting evidence for their validation. It should still be noted that the safest theoretical claim to be made based on evidence from (spontaneous) speech error data can concern only the characterisation of occurring errors and the implications they invite.

2.1.4.1 Types of Speech Error

Speech errors can be described along two dimensions, the size of the unit involved in the error (e.g. phrase, word, morpheme, syllable, phoneme etc.) and the nature of the error itself (e.g. addition, omission, substitution). This type of classification yields many different categories. Table 2.1 gives some examples of speech errors classified in this way.

The error is said to be *contextual* if a source can be found from within the utterance, and *non-contextual* if it cannot. The types of errors with which I shall be concerned are the errors involving contextual slips or movements of phonemes. These are contextual additions and contextual substitutions which coarsely form three broad classes of error type relevant to the serial order of phonemes in speech production. They are characterised at the segment level as exchanges, anticipations and perseverations. They arise either by the replacement or addition of a phoneme from elsewhere in the context of the utterance and may be described in terms of two

features; the erroneous segment and its location, and the segment that is said to have caused the error.

Table 2.1.

Examples of speech errors from Harley and MacAndrew's (1995) corpus, classified by the size of unit and the nature of the error.

Error type	Utterance	Target
phoneme deletion	I've di_covered the secret	discovered
phoneme substitution	a famous **v**allerina	**b**allerina
phoneme anticipation	Do **fr**ies have **br**ains?	**fl**ies
phoneme perseveration	a**t**omic **w**oppens	**w**eapons
phoneme exchange	Do you **r**eel **f**eally bad?	**f**eel **r**eally bad
phoneme addition	the worst **p**lossible revenge	possible
affix deletion	the chimney catch_ fire	catch**es**
morpheme exchange	**cook**en **chick**ed in beer	**chick**en **cook**ed in beer
syllable deletion	you have no re-lection	re**col**lection
word substitution	give me a **fork**	**spoon**
word exchange	Guess whose **mind** came to **name**?	whose **name** came to **mind**
word blend	the **chung** of today	children + young
phrase blend	miss you **a very much**	a great deal + very much

Anticipation errors are the most common type of movement errors and are characterised by a segment appearing too soon, as in [2.1].

[2.1] John dropped his cu**ff** of coffee -> cup

Perseveration errors are less common than anticipations and are characterised by a segment appearing again later than originally planned, as in [2.2].

[2.2] John gave the goy -> boy

The exchange error is more complex, and there are fewer exchanges observed than perseverations. Exchanges are characterised by the transposition of a pair of phonemes, as in [2.3].

[2.3] teep a cape -> keep a tape

Errors [2.1] through [2.3] are from Fromkin (1971).

The analysis of error corpora, like hesitation analysis, provides evidence for the existence of units of speech (Fromkin, 1971) and the levels of processing at which they are manipulated (Fromkin, 1971; Garrett, 1975). Many properties have been observed from error corpora concerning the size and nature of interacting units in errors. For example, errors tend to obey unit categories in that words interact with words, phonemes with other phonemes and so on. Even within the unit categories, class type is also obeyed in interacting error components: content words (words that convey semantic information about a concept) replace content words, vowels replace vowels and consonants replace other consonants. It is extremely rare for a vowel to replace a consonant or a function word (words that convey grammatical information) to replace a content word for example.

2.1.4.2 Constraints on Sound Movement Errors

In this section, discussion is confined to the constraints on sound movement errors only, since errors involving larger units are not directly informative as to the problem of serial order of speech sounds. Similarities between interacting components of sound errors include the constituent syllabic position of phonemes, their featural similarity, the distance they span and the fact that the erroneous result of any slip is almost always a pronounceable sequence (i.e. it does not violate the phonotactic rules of the language). These apparent patterns in the speech error data suggest that the errors are constrained in certain ways.

Sound exchange errors are more likely to occur when the sounds that exchange are relatively close together. That is, they are constrained by distance (Boomer & Laver, 1968; Broeke & Goldstein, 1980; Garrett, 1975; MacKay 1970). Sound exchanges nearly always occur within the same surface clause. This is not necessarily seen for errors involving units larger than sounds. For example, whole word exchanges can span a much greater distance, even across surface clauses. For sound errors (specifically exchanges), this is known as the *distance constraint*, and is easily illustrated by a plot of the frequency of exchange errors as a function of the distance in between the transposed phonemes. MacKay (1970) analysed an early collection of sound exchange errors (Meringer & Mayer, 1895) which clearly shows the distance constraint. This is illustrated in Figures 2.1 and 2.2. Figure 2.1 illustrates the constraint for exchange errors occurring within a word and Figure 2.2 shows the constraint is also apparent when sound exchanges cross word boundaries.

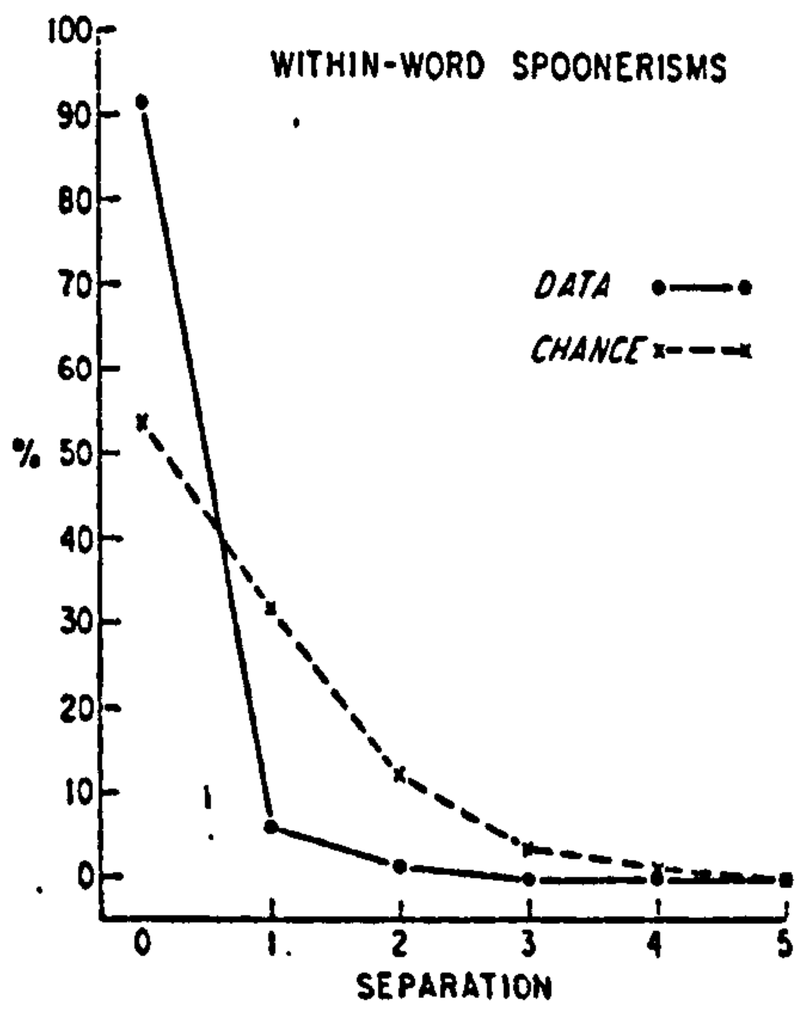

Figure 2.1. The proximity of within-word consonant exchanges. Chance is calculated as the possible frequency of exchanges in words containing the exchange errors, assuming that they occur at random. (Reproduced from MacKay, 1970.)

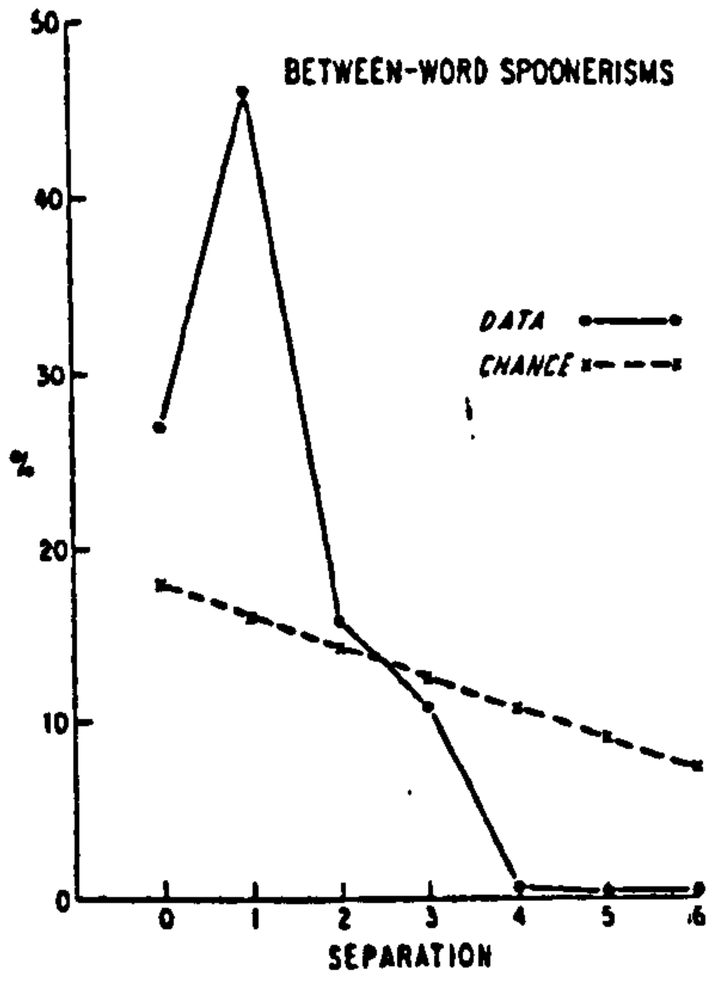

Figure 2.2. The proximity of between-word consonant exchanges. Chance is calculated as the possibly frequency of exchanges in sentences of that length, assuming that they occur at random. (Reproduced from MacKay, 1970.)

The phonological structure of the speech stream plays a role in preventing certain types of error. The relative syllabic position of phoneme movement errors is usually preserved (MacKay, 1970; Shattuck-Hufnagel, 1979). That is, when phonemes are involved in an exchange, both phonemes occupy the same syllabic position in the phonological structure. This is the *syllable position* constraint. For example, exchanges like [2.4] (from Fromkin, 1971) in which both phonemes are in the onset of a syllable are quite common, whereas an exchange like [2.5] where one phoneme is in the onset and the other in the coda of a syllable, is quite rare.

[2.4] sood sherve -> should serve

[2.5] teg -> get

Particular parts of the phonological structure also appear to influence error patterns. For example, syllable-initial and word-initial consonants are more frequently involved in exchanges than syllable-final or word-final consonants (Broeke & Goldstein, 1980; MacKay, 1970). This is the *initialness* constraint.

The phonetic form of sounds influences their chances of becoming confused or replaced by an alternative similar sound, and this is seen in the *phonetic similarity* constraint and the *phonotactic* constraint in three ways. First, exchanging phonemes very often belong to the same class (i.e. consonant or vowel) and very rarely exchange with sounds from a different class. In other words consonants exchange with other consonants and vice versa for vowels, and consonants hardly ever exchange with vowels. Second, the sounds that exchange are usually very similar to each other in their phonetic or featural description (Broeke & Goldstein, 1980; Ellis, 1980; MacKay, 1970; Shattuck-Hufnagel, 1979; Shattuck-Hufnagel & Klatt, 1979). The phonetic similarity constraint can be clearly seen in Figure 2.3 which again shows data from Meringer and Mayer's (1895) error corpus, as analysed by MacKay (1970).

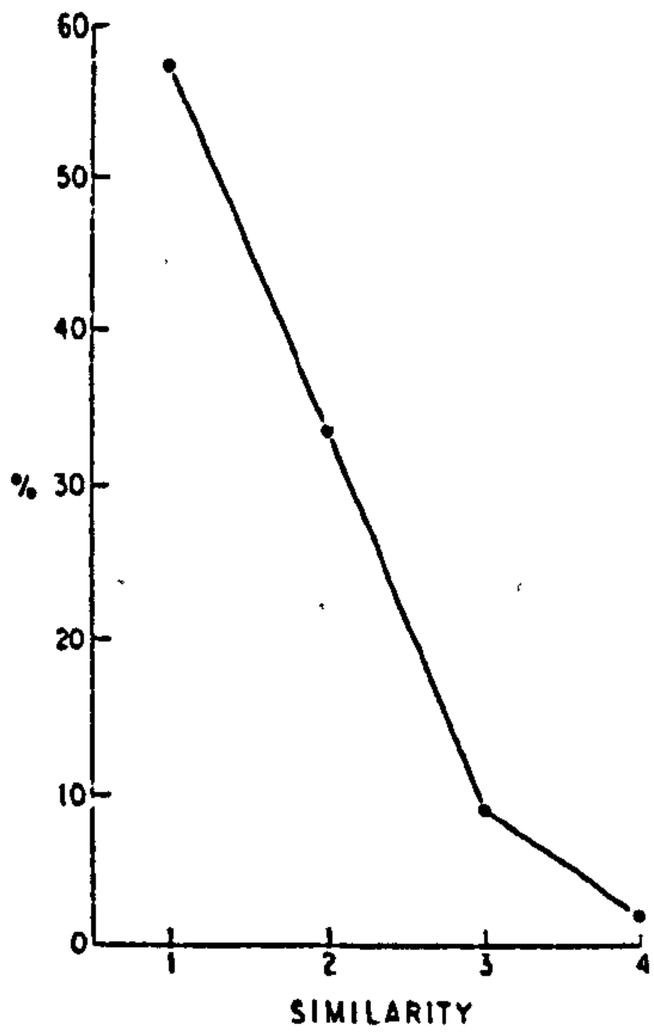

Figure 2.3. Frequency of single-phoneme consonant exchange errors as a function of similarity of the exchanging phonemes, based on the number of different distinctive features. (Reproduced from MacKay, 1970.)

Third, errors that would cause a violation of the phonotactic rules of the language very rarely occur. It is a well observed pattern that phoneme errors always result in

pronounceable sequences that do not violate the phonotactic constraints of their language (Boomer & Laver, 1968; Fromkin, 1971; Shattuck-Hufnagel, 1979; MacKay, 1970). This effect was so strong that Wells (1951, as cited in Fromkin, 1971) stated it as his first Law of tongue slips as 'A slip of the tongue is practically always a phonetically possible noise'. Somewhat related to the phonotactic constraint is the *environment* constraint. The similarity of the immediate environment of exchanging phonemes is important; the likelihood of an exchange between two phonemes is increased if the phonemes before or after them are the same (Wickelgren, 1969; MacKay, 1970; Ellis, 1980). This constraint concerns the similarity of the immediate environment in which the error occurs, rather than the actual erroneous segment itself.

This concludes the brief review of some of the research methods used in the investigation of speech production. The patterns and constraints on speech errors have been instrumental in providing evidence for theories concerning serial ordering during phonological encoding. They have also been a useful tool in the research on lexicalisation (the process of verbalising internal thoughts), both of which I shall return to in the next chapter.

2.2 An Analysis of Sound Movement Errors

In this section I present a more precise and comprehensive characterisation of all the sound movement errors found in Harley and MacAndrew's (1995) corpus of speech errors.

2.2.1 The Corpus

The analyses in this section are based on phoneme errors from Harley and MacAndrew's (1995) corpus. Speech errors were collected by Harley at the Universities of Cambridge, Cardiff, Dundee and Warwick over a period of several

years. All errors were made by adult, native English speakers. Every error made in a series of predetermined periods was noted and recorded as soon as possible after its occurrence with as much of the preceding context and conversation as possible. The speaker was immediately interviewed to establish the context of the error. This included the speaker's thoughts at the time of the error, the intended target, any environmental context of importance, and what the speaker thought the cause of the error was. (See Harley, 1984, for an initial description of the corpus, method, and preliminary findings.) This stringent collection method has the advantage that the effects of biases mentioned in the previous section are minimised. The total number of errors in the corpus is currently 6753.

2.2.2 Method

One of the main problems in analysing error data is that of classification. The context is a helpful aid in this task, although some errors remain ambiguous even in the light of a rich context. The types of errors I shall be concerned with are the errors involving contextual slips or movements of phonemes. I shall refer to these errors as *contextual sound errors*, or *sound movement errors*. Four broad classes of error type are relevant here to the serial order of phonemes in speech production. They are characterised at the segment level as exchanges, migrations, anticipations and perseverations. They arise either by the movement of phonemes within an utterance or by the replacement or addition of a phoneme from elsewhere in the context of the utterance and consist of two components; the locus and the source. I will refer to the *locus* (i.e. the erroneous segment and its location) as the segment which has unintentionally either been added into the utterance or replaced another segment in the utterance and the *source* of the error as the segment from elsewhere in the utterance that is the same as the erroneous intrusion. A guide to the symbol notation used to represent phonemes can be found in Appendix III.

2.2.2.1 Classification

2.2.2.1.1 Anticipations

Anticipation errors arise when a phoneme in the utterance is accidentally produced prematurely. The anticipated phoneme can either appear as an additional phoneme somewhere to soon (anticipation by addition, as in [2.6]), or can replace an earlier phoneme (anticipation by substitution, as in [2.7]).

[2.6] King's memblers only -> King's members only

[2.7] autowatic washing machine -> automatic washing machine

The phrases on the left contain the speech error and the phrase on the right is the intended utterance. The phoneme involved in the error has been highlighted in bold type. In the case of anticipation errors, the first phoneme in bold type indicates the locus of the anticipation and the second indicates the source of the error. In [2.6] the locus is thus /l/ in 'memblers' and the source is /l/ in 'only'.

2.2.2.1.2 Perseverations

Perseverations arise in a similar way when a phoneme already produced in the utterance is unintentionally repeated at a later time. The perseverated phoneme can either appear as an additional phoneme within the utterance (perseveration by addition, as in [2.8]) or can replace a phoneme later on in the utterance (perseveration by substitution, as in [2.9]).

[2.8] then Scotland would squalify -> then Scotland would qualify

[2.9] is it fish and **chip ch**ime yet? -> is it fish and chip time yet?

2.2.2.1.3 Migration

Migration errors involve the movement of a phoneme from its target position to elsewhere in the utterance. Migration errors can either move forward in the utterance (perseveratory migration, as in [2.10]) or backwards (anticipatory migration, as in [2.11]).

[2.10] champagne-f_avoured toothplaste -> champagne-flavoured toothpaste

[2.11] too much to flit on my p_ate -> too much to fit on my plate

2.2.2.1.4 Exchanges

Finally, *exchange* errors involve the movement of two phonemes in the utterance. The basic feature of an exchange is two phonemes swapping places, as in [2.12]. Exchanges can also be viewed as the result of an anticipation (by substitution) complemented with a perseveration (by substitution) of the phoneme displaced by the anticipation, which then replaces the source of the anticipation.

[2.12] **p**ick as a sarrot -> sick as a parrot

2.2.2.1.5 Ambiguous migration/exchanges

Some of the migration errors were placed in a separate category because they could not be discriminated between either migration or exchange errors. This category was called 'ambiguous migration/exchanges' and contained errors such as those in [2.13].

[2.13] a lot of unem**p**oyed **pl**eople -> a lot of unemployed people

This error could be classified either way as a migration or an exchange because a migration of the /l/ from 'unemp_oyed' to '**pl**eople' and an exchange of the consonant cluster /pl/ in 'unem**pl**oyed' and the consonant /p/ in '**p**eople' result in the same utterance and which of the two error mechanisms is responsible remains opaque.

The analyses that follow are based on speech errors from Harley and MacAndrew's (1995) corpus that fell into the categories of exchanges, migrations, anticipations and perseverations at the segment level. Errors that could fall into more than one category (i.e. either anticipation or perseveration) were classed as ambiguous errors (as in [2.14]) and were excluded from the analyses.

[2.14] **b**ut **b**arkling water is **b**ad for you -> but sparkling water is bad for you

Although it is traditional to assume the closest contextual phoneme to be the source of the error in cases of ambiguity (so the error in [2.14] would be classified as a perseveration from /b/ in "**b**ut") there is no reason to suppose that other contextual phonemes can be ignored (i.e. the /b/ in "**b**ad"). The error in [2.14] could equally as well have been primed by the /b/ in "**b**ad", or even a combination of both instances of /b/ in "**b**ut" and "**b**ad". The ambiguous errors formed their own category and were analysed separately.

Errors where more than type of error had occurred at the same time were classed as complex errors, such as [2.15] where an exchange and an omission have occurred, and were also omitted from the analyses.

[2.15] **grep**_ant -> pregnant

A total of 2289 sound movement errors were categorised according to the above descriptions. This method of classification resulted in a total of 861 anticipations (37.6%), 643 perseverations (28.1%), 241 exchanges (10.5%), and 321 non-contextual substitutions (14.0%). Other errors (30 ambiguous exchange/migrations, 25 migrations, 155 ambiguous errors, 13 complex) accounted for the remaining 9.7% of errors analysed, as illustrated in Figure 2.4.

Before a quantitative analysis of the errors is discussed, a brief reminder of certain qualitative properties observed in other corpora is given. The basic properties are first summarised.

1. *Distance Constraint.* Sound exchange errors are constrained by distance (Boomer & Laver, 1968; Broeke & Goldstein, 1980; Garrett, 1975; MacKay 1970). Sound errors nearly always occur within the same surface clause whereas whole word exchanges can span a greater distance.

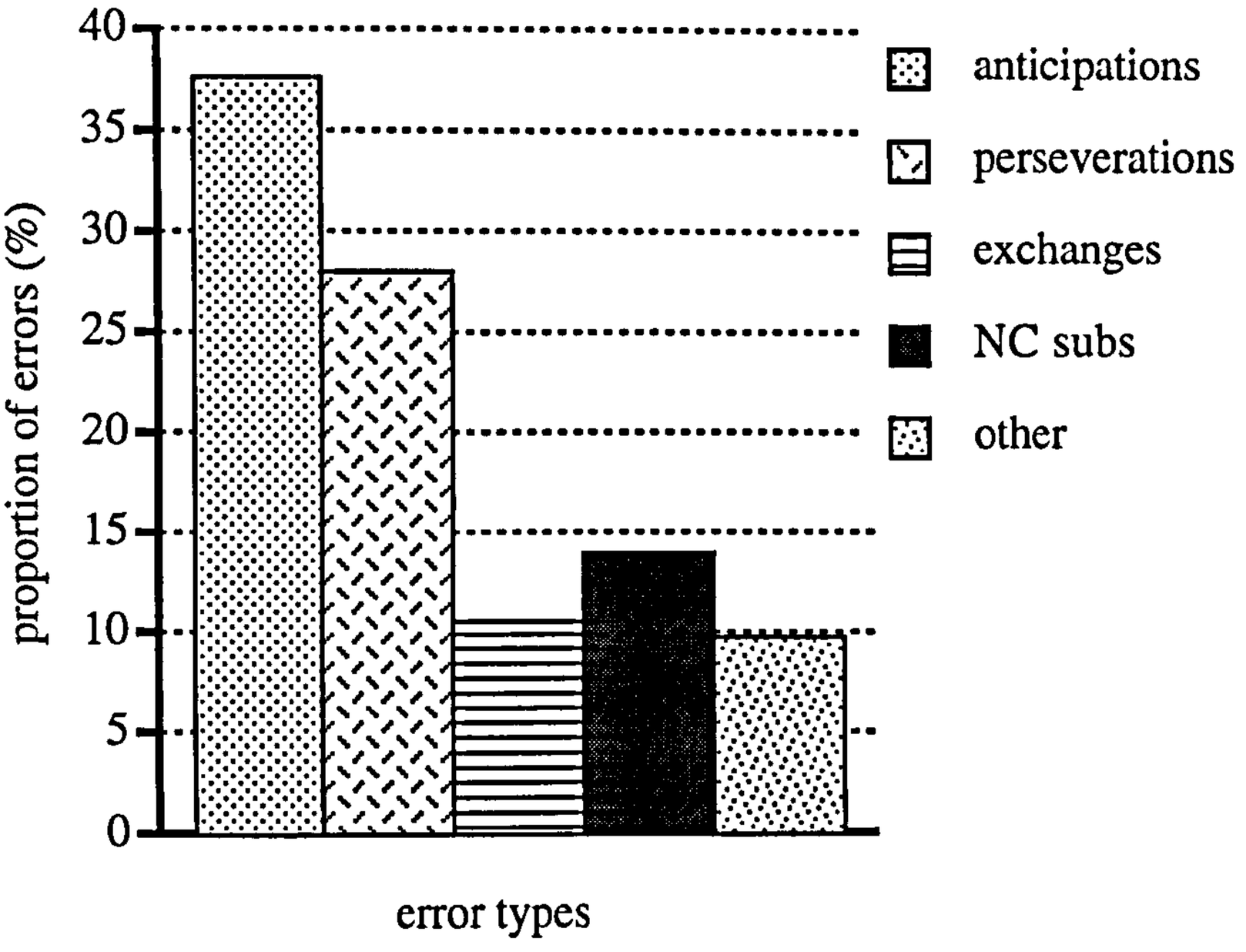

Figure 2.4. Distribution of error types (N=2289) of sound errors from Harley and MacAndrew's (1995) corpus. (NC subs = non-contextual substitutions.)

2. *Syllable Position Constraint.* The syllabic position of phoneme movement errors is usually preserved (MacKay, 1970; Shattuck-Hufnagel, 1979). When phonemes are involved in an exchange, they both occur in the same syllabic position such that exchanges like *fyrostoam -> styrofoam* occur frequently whereas *teg -> get* do not usually occur.

3. *Phonetic Similarity Constraint.* Exchanging phonemes are usually similar to each other in their phonetic or featural description (Broeke & Goldstein, 1980; Ellis, 1980; MacKay, 1970; Shattuck-Hufnagel, 1979; Shattuck-Hufnagel & Klatt, 1979). Thus consonants only ever exchange with other consonants and vice versa for vowels, and consonants hardly ever exchange with vowels.

4. *Environment Constraint.* The similarity of the immediate environment of exchanging phonemes is also important; the likelihood of an exchange

between two phonemes is increased if the phonemes before or after them are the same (Ellis, 1980; MacKay, 1970; Wickelgren, 1969).

5. *Initialness Constraint.* Syllable-initial and word-initial consonants are more frequently involved in exchanges than syllable-final or word-final consonants (Broeke & Goldstein, 1980; MacKay, 1970).

6. *Phonotactic Constraint.* Phoneme errors always result in pronounceable sequences that do not violate the phonotactic constraints of their language. (Boomer & Laver, 1968; Fromkin, 1971; MacKay, 1970; Shattuck-Hufnagel, 1979). Hence exchanges such as *dnurk* -> *drunk* almost never occur.

The quantitative analysis of the errors from Harley and MacAndrew's (1995) corpus, based on these observed patterns is now described.

2.2.3 Results

The speech errors were subjected to five analyses with respect to similarity patterns between the source and intruding phoneme (where exchanges are concerned, both phonemes are source and locus, although for the sake of some of the analyses, the first location of the exchange is referred to as the locus of the error, and the second, the source) in different dimensions. The similarities under consideration are proximity (in terms of syllable separation), phonetic feature composition, within-syllable position, surrounding environment and the presence of lexical stress.

2.2.3.1 Proximity

All the errors were analysed with respect to the proximity of the source and intruding phoneme in terms of the number of intervening syllables between them. MacKay (1970) suggested that all exchange errors whose source and intruding phoneme appeared in adjacent syllables had a separation of 0, a separation of 1

reflected one intervening syllable between source and locus and so on. This method is used to assign separation to the errors in this analysis. The separation as measured for the exchange errors is between the two phonemes involved in the exchange, since either could be the source or locus. Consonants and vowels were analysed separately, as were within-word and between-word exchanges. Results in figures and tables are expressed as percentages unless otherwise specified.

Figure 2.5 shows the proximity of exchanges in within-word consonant exchanges.

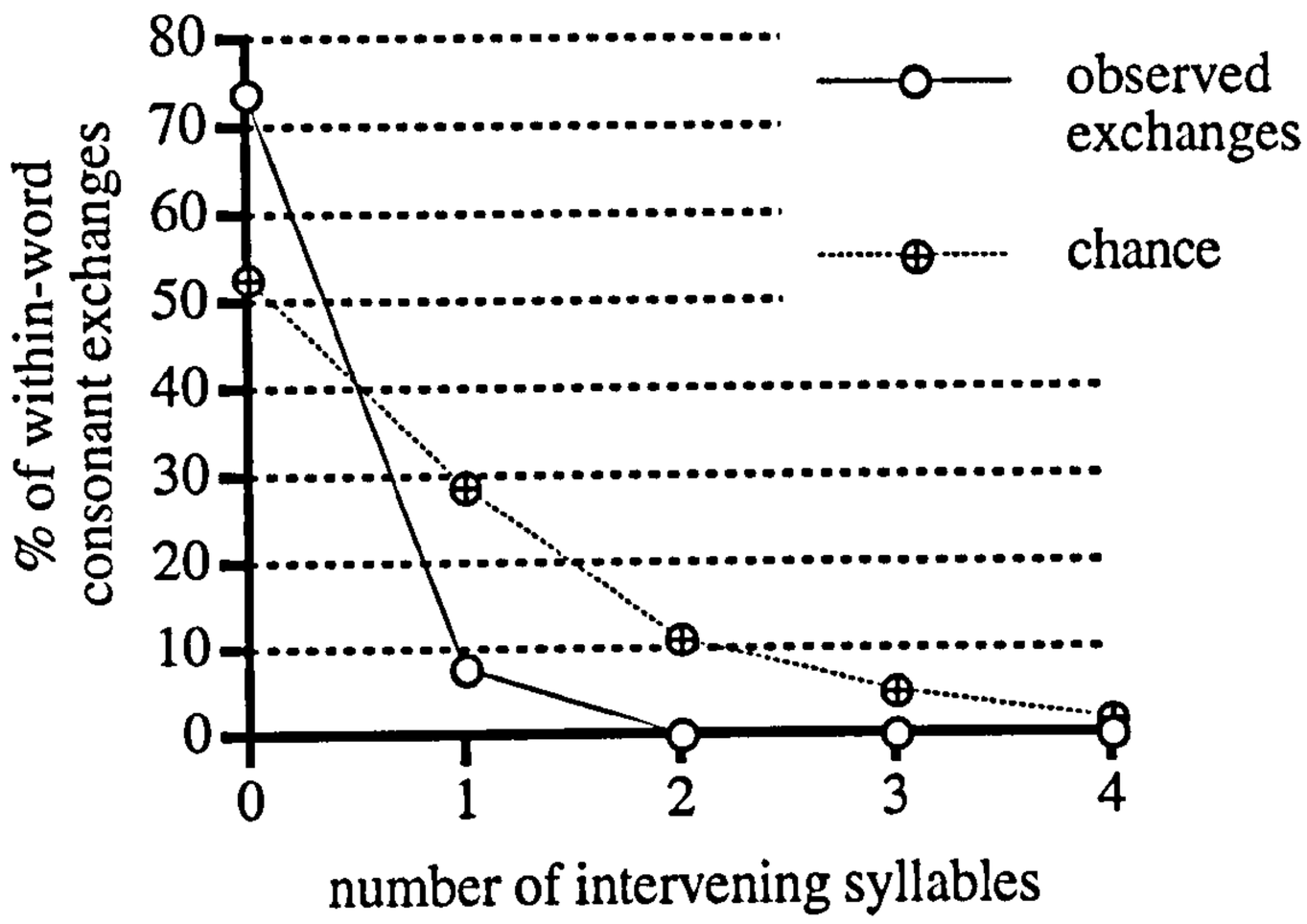

Figure 2.5. Proximity of within-word consonant exchanges (N=50).

In order to comment on the statistical significance of these results and their relevance it is necessary to have some estimation of the null hypothesis, which here is the expected proximity of reversed phonemes if reversals occur by chance. MacKay (1970) described a method of calculating the chance probability of exchanges occurring at different degrees of separation, based on the assumption that any pair of phonemes can exchange at any point within the word. However, calculation based on this assumption may not be the most realistic estimation. Consider for example the

word "escapade" (*/e s k a p eI d/*). If any pair of phonemes can be exchanged, then presumably sequences such as */k s e a p eI d/* or */e s k d p eI a/* would be possible, which violate phonotactic constraints. Thus by including these exchanges as possible chance occurrences in the calculation of the null hypothesis, an unrealistic estimation may be obtained.

For between-word exchanges a similar procedure (MacKay, 1970) estimates the chance proximity based on the assumption that exchanges occur at random. The estimation is based on the number of syllables per phrase or sentence (average length 7 syllables), whichever was recorded in the corpus.

Figure 2.6 shows the proximity of between-word exchanges compared to the chance proximity, and Table 2.2 shows the raw frequencies of separation for within-word and between-word consonant exchanges.

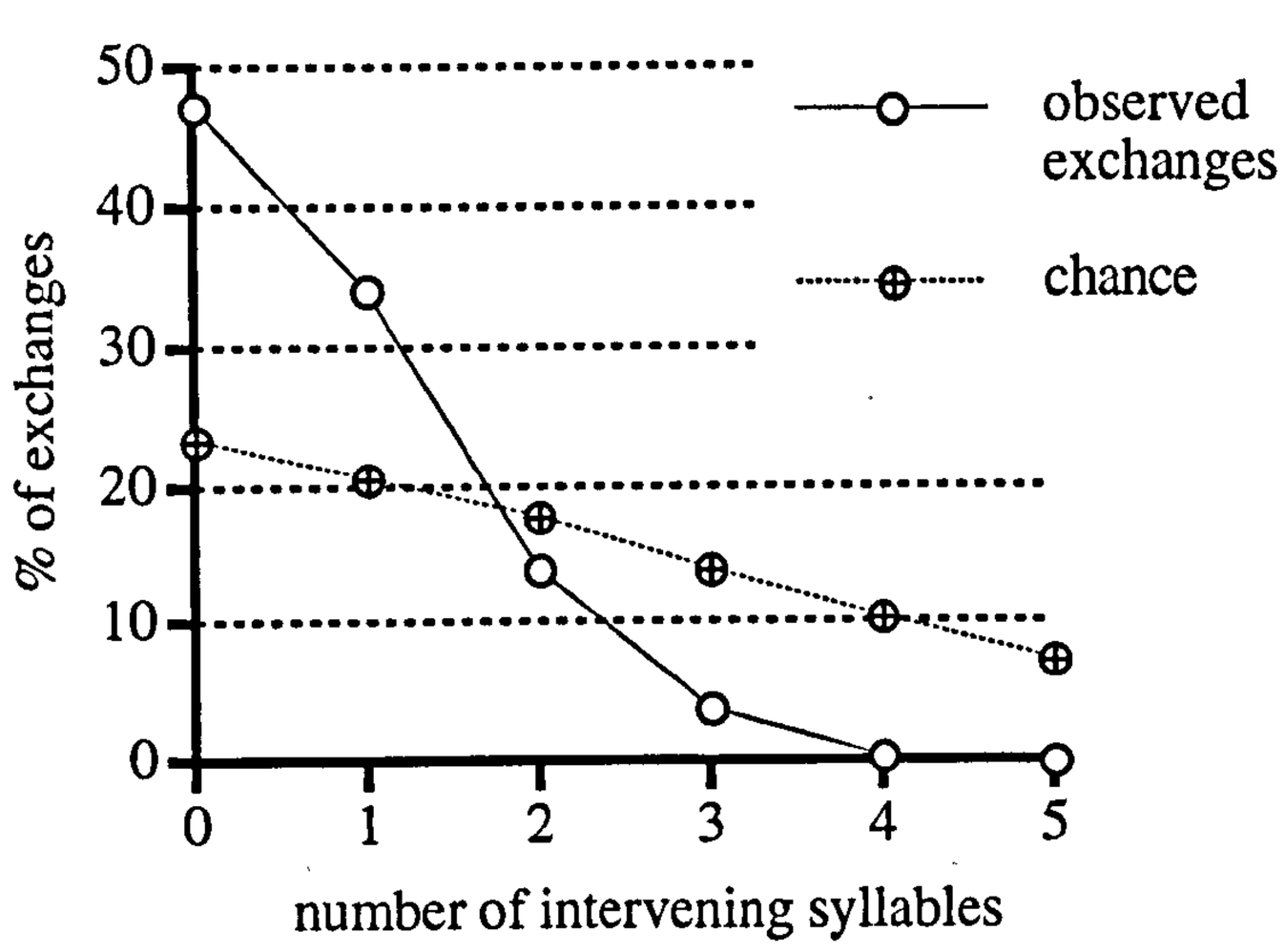

Figure 2.6. Proximity of between-word consonant exchanges (N=173).

The same caution must be made for between-word consonant exchanges as was made for within-word consonant exchanges in that the estimation of exchanges occurring by chance may not be representative for reasons already stated. A

chi-squared test on the frequency of consonant exchange separations was highly significant (χ^2=68.47, df=5, p<0.001), showing that the distribution of phoneme separations in the between-word exchange errors was significantly different from chance.

Table 2.2.

Raw frequencies of separation for within-word and between-word consonant exchanges, for data and chance values.

error type	separation					
	0	1	2	3	4	5
within-word	37	4	0	0	0	0
chance	97	53	21	9	3	0
between-word	82	59	24	7	2	0
chance	80	72	61	49	37	25

When taken together as if word boundaries are irrelevant, the proximity function for within-word and between-word consonant exchanges shows a smooth non-linear monotonic decrease in separation, as can be seen in Figure 2.7.

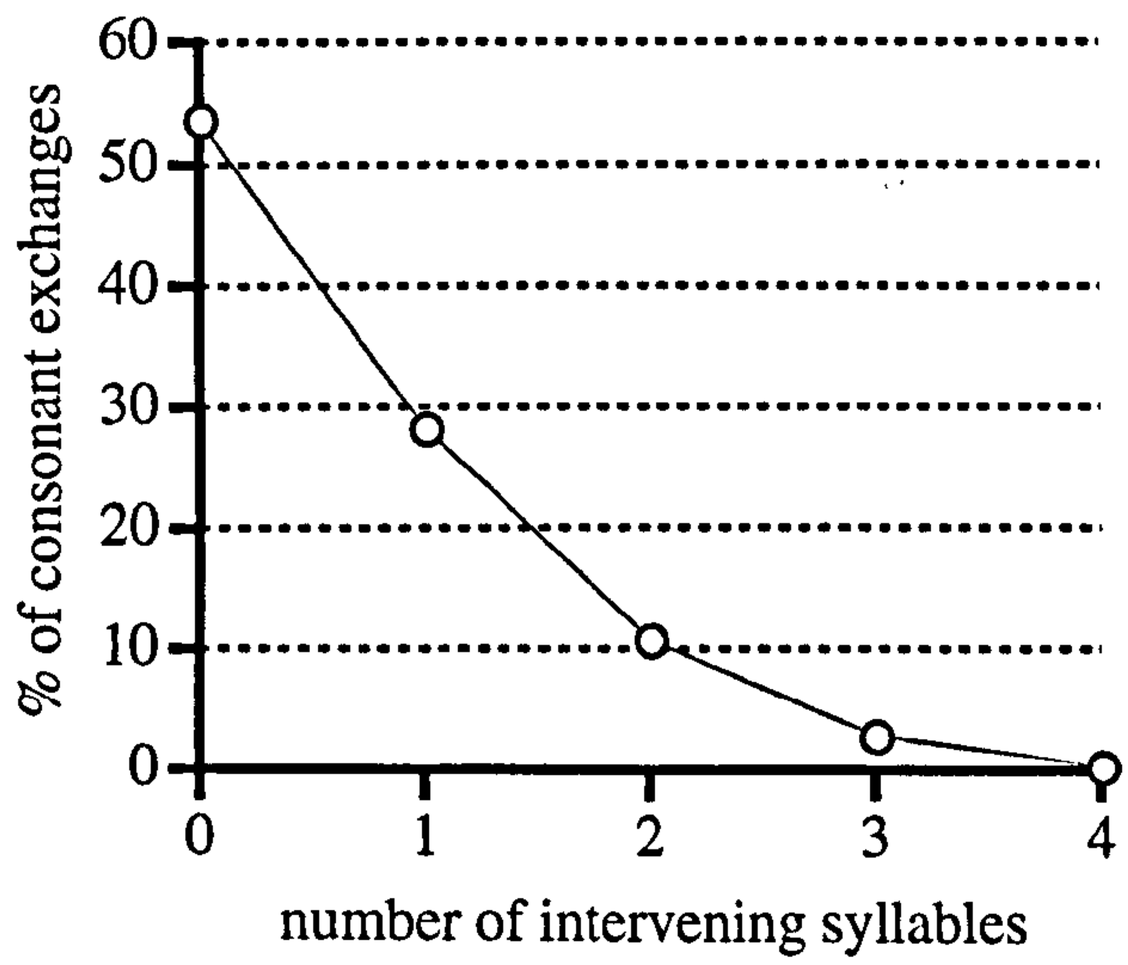

Figure 2.7. Proximity of between-word and within-word consonant exchanges (N=223).

The pattern for exchanging vowels is less clear. Between-word and within-word categories were collapsed for vowel exchanges because there was only one within-word vowel exchange, in [2.16].

[2.16] Harry Radknepp -> Harry Redknapp

Figure 2.8 shows the proximity function for the vowel exchanges. The results show a greater tendency for vowel exchanges to occur with a 1 separation. This contrasts with consonant exchanges, in which phonemes were most likely to exchange at 0 separation. However there were very few vowel exchanges in comparison to consonant exchanges (N=18 for vowels, N=223 for consonants) which makes it difficult to interpret the general pattern of the vowel data from so few points.

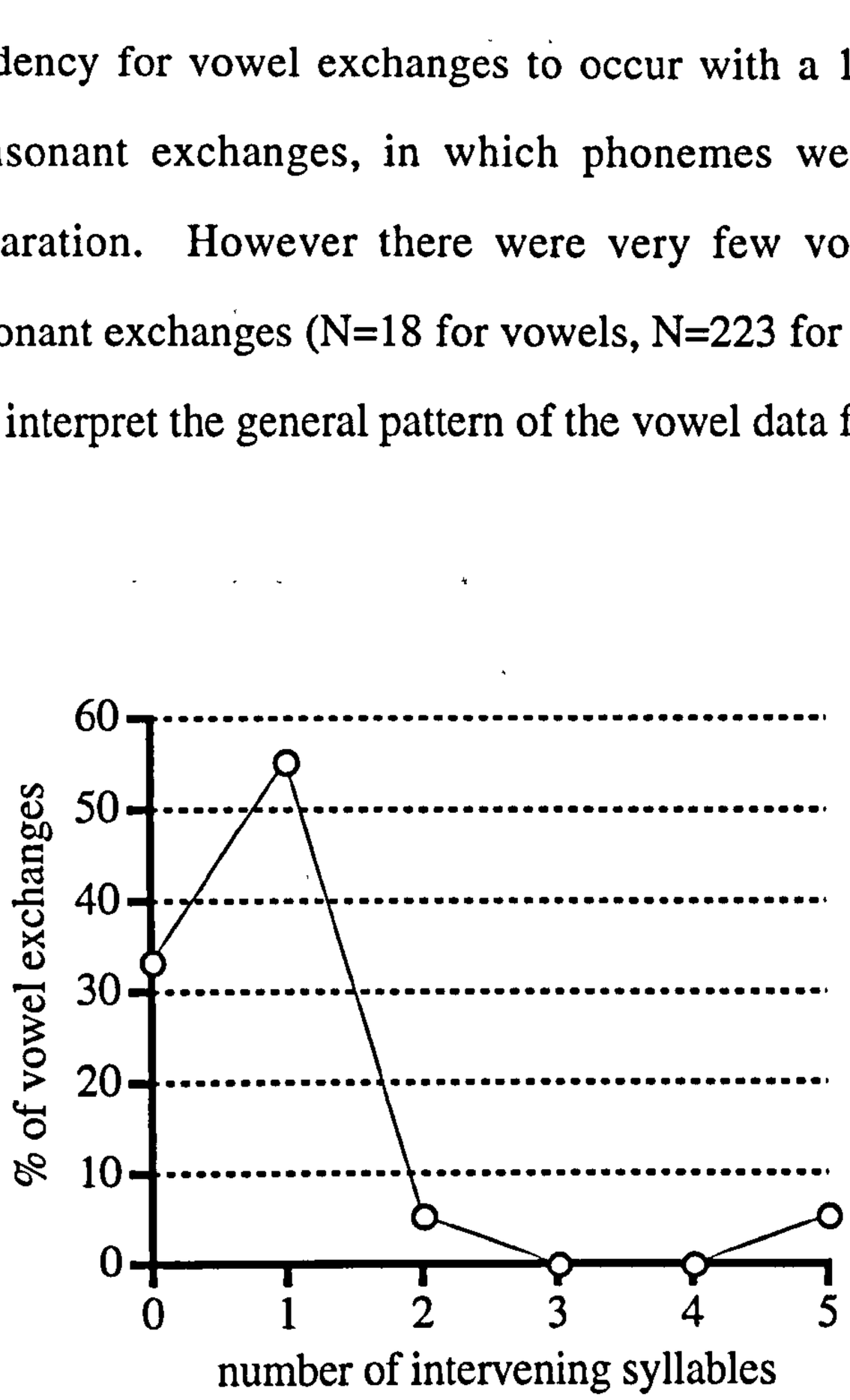

Figure 2.8. Proximity of between-word and within-word vowel exchanges (N=18).

The proximity functions for the migration errors and the ambiguous exchange/migration errors yielded similar results, although in both cases there were not many occurrences (N=25 and N=30, respectively). In all cases, source and locus were never separated by more than 2 syllables. Figure 2.9 shows how the proximity

of migration errors compares to the ambiguous errors and to the between-word consonant exchanges. As can be seen from the graph, both migration errors and ambiguous migration/exchange errors exhibit a proximity function very similar to consonant exchanges.

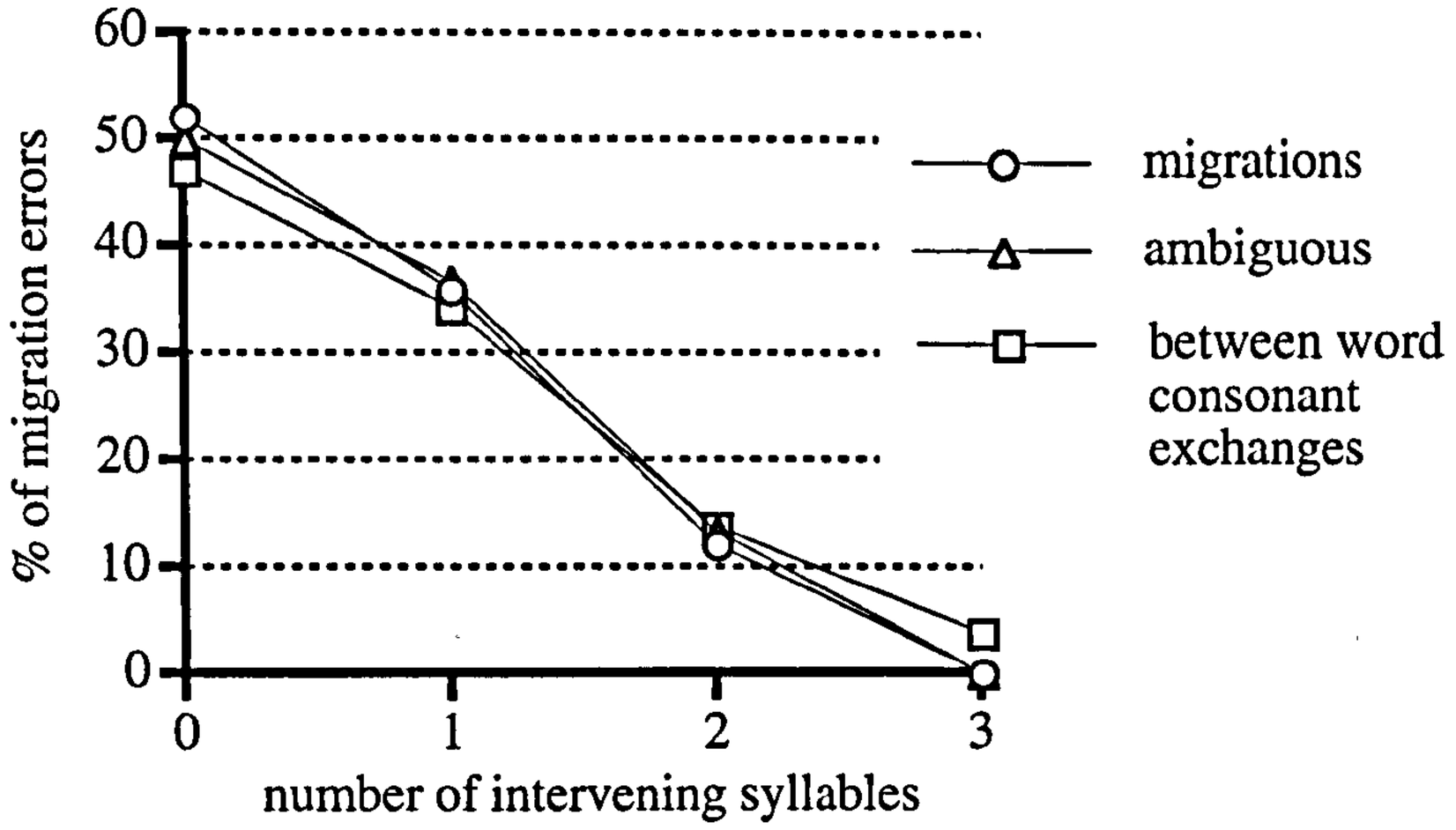

Figure 2.9. Comparison of proximity of migrations (N=25), ambiguous migration/consonant exchanges (N=30) and between-word consonant exchange errors (N=173). (Separation of exchange errors has only been shown in the figure for a separation up to 3 syllables for a clearer comparison to the migratory and ambiguous errors.)

The proximity functions for anticipatory and perseveratory errors show that errors involving only one segment (as opposed to two in exchange errors) can occur over a longer distance. Figure 2.10 shows the proximity functions for all consonant anticipation and perseveration errors.

The following errors are examples of some longer range anticipations and account for all the errors whose separation was greater than six syllables.

[2.17] gives ones **l**eyes a chance to adapt to the **l**ight -> ... eyes ...

[2.18] a **p**rity there are so many bones in **R**ed mullet -> ... pity ...

[2.19] I'm quite **p**reased with myself for having convinced **R**eeves -> ... pleased

The example in [2.17] has been anticipated by seven syllables, and [2.18] and [2.19] by eight syllables.

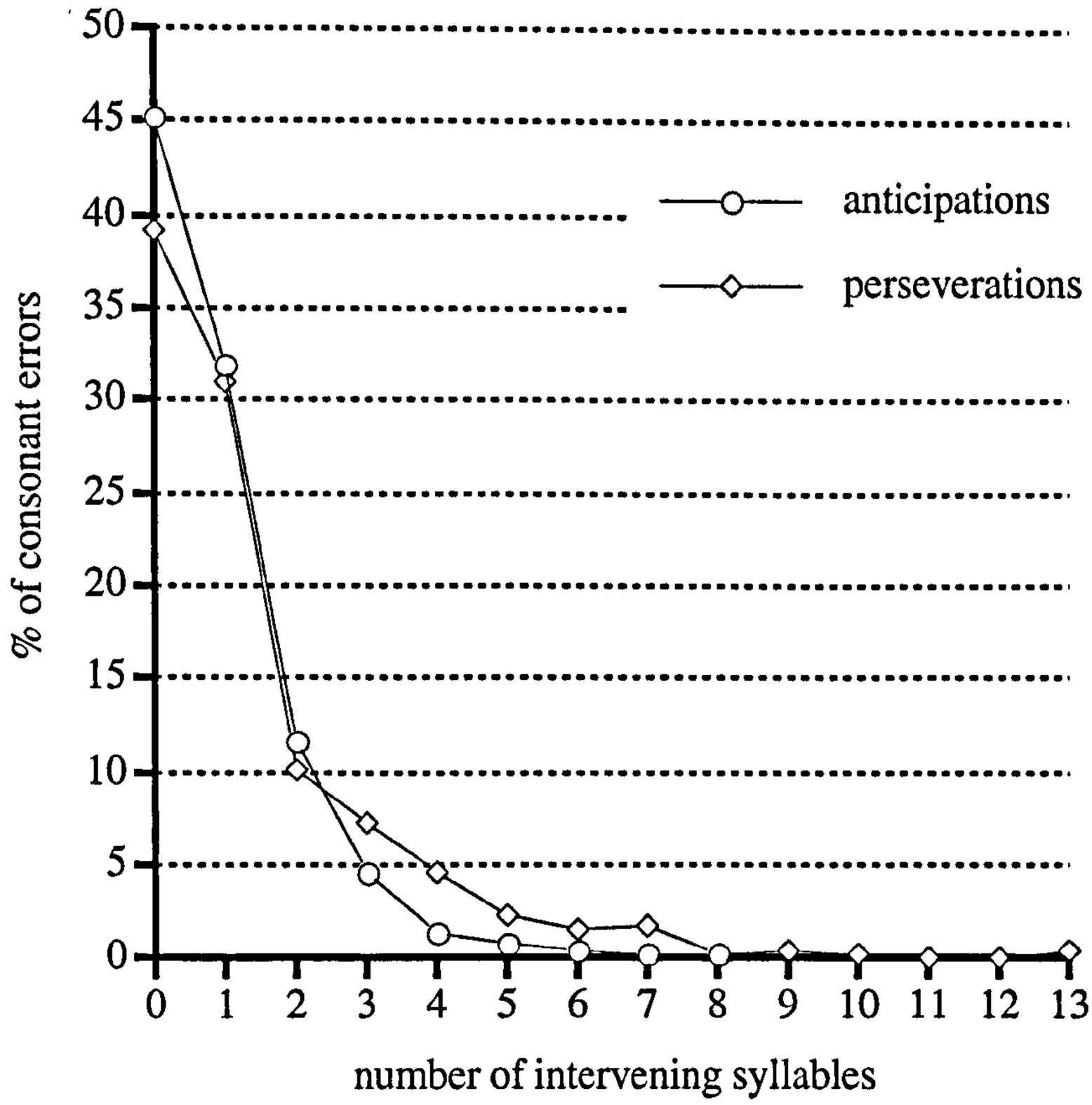

Figure 2.10. Proximity of locus and source of consonant anticipation (N=733) and perseveration (N=534) errors.

The examples in [2.20] to [2.22] account for all the perseveration errors greater than nine syllables apart. In [2.20] the separation between source and locus is 10 syllables, and in [2.21] and [2.22] it is 13.

[2.20] Are we going to st**ick** to four three three, or are we going to sw**ick** to four two two? -> ... switch ..

[2.21] It's the se**qu**el to the Andromeda strain - what's the Andromeda s**qu**ain about? -> ... strain ...

[2.22] I'll sit down an**d** contemplate the ultimate nature of reali**d**y -> ... reality

The anticipatory and perseveratory movement of vowels also occurred over a greater distance than vowel exchanges, as can be seen from the proximity function for vowel anticipations and perseverations in Figure 2.11.

For all the proximity analyses, for all error types, the source and locus of most errors occur within three syllables of each other. The average separation of exchanging consonants (0.63) is less than for exchanging vowels (0.96). That is the locus and source of exchanging consonants are on average closer together than vowel exchanges. This is also true of anticipation errors (average separation of 0.8 for consonants and 1.2 for vowels), but not for perseverations (average separation of 1.3 for consonants and 1.2 for vowels).

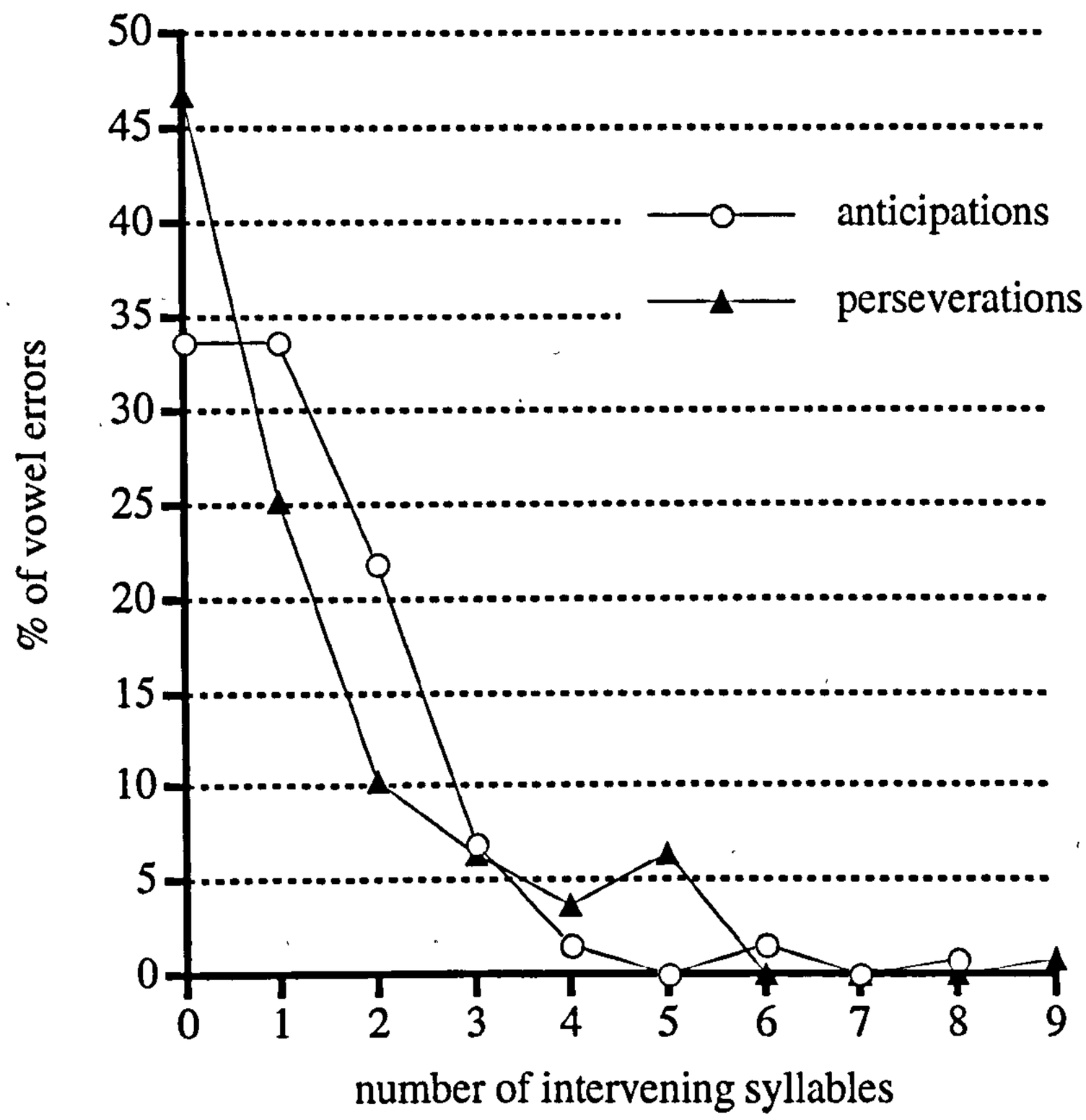

Figure 2.11. Proximity of locus and source of vowel anticipation (N=128) and perseveration (N=107) errors.

On average, error segments in exchange errors are closer together than anticipation error segments, which in turn are closer together than perseveration error segments. Qualitatively, all the movement errors show an increased tendency to involve nearby rather than distant segments in the utterance. It is possible, however, that the anticipation and perseveration errors that occur over a greater distance occur by chance, because the further the search is extended over the utterance to find a source, the more likely one is to be found. Long distance anticipation and perseveration errors could be estimated if the chance distribution of error separation was estimated, as it was for exchange errors. If the long distance anticipation and perseveration errors were classed as outliers, then it may be the case that the differences in the distributions of error types would no longer be apparent, since over a range of 3 or 4 syllable separation, all error types show a similar distribution and thus the average separation for all error types may show less of a difference between types.

2.2.3.2 Phonetic Similarity

All exchange errors (between-word and within-word collapsed) involving single consonants were analysed with respect to their phonetic similarity. Phonetic similarity was determined according to Wickelgren's (1966) distinctive feature system. By this specification, phonemes are defined along four dimensions: place of articulation, voicing, nasality and manner of articulation. Similarity is thus scored by the number of dimensions along which a pair of phonemes differ. Hence a low similarity score indicates two phonemes are highly similar, and a high score indicates they are dissimilar. For example, /k/ and /g/ differ only in voicing and therefore have a similarity of one (i.e. very similar), whereas /f/ and /g/ differ in voicing, manner and place of articulation and therefore have a similarity of 3 (i.e. not similar). Figure 2.12 shows the results of the similarity analysis of exchanging consonants as scored using Wickelgren's (1966) distinctive feature scheme.

The phonetic similarity analysis is only relevant to single consonant exchanges and not to migration, anticipation or perseveration errors because the latter error types only involve one segment type.

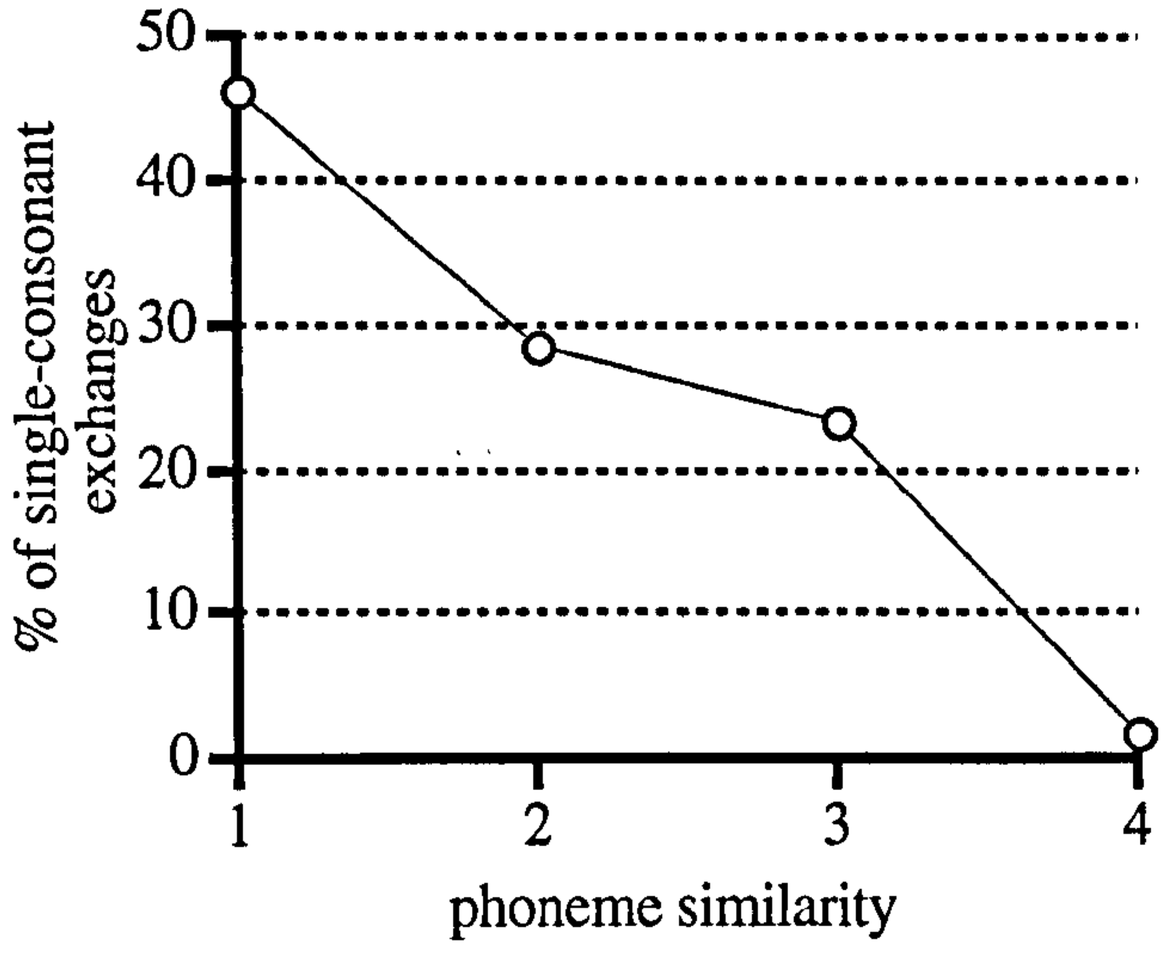

Figure 2.12. Phonetic similarity (i.e. the number of different distinctive features) of single consonant exchanges (between-words and within-words, N=171).

Figure 2.12 shows a strong tendency for exchanging phonemes to occur if they are phonetically similar. Thus phonetically similar consonant pairs such as /m/ and /n/ or /k/ and /g/ are much more likely to exchange than dissimilar pairs such as /f/ and /g/.

2.2.3.3 Syllabic Similarity

All the errors were analysed in terms of their syllabic position. Consonants either appear in the onset or the coda of a syllable (either as a single phoneme or a cluster) and vowels occupy one of three possible syllabic positions; syllable-initial as in "age", medial as in "cut" or syllable-final as in "bee". If the locus and source of the error occupied the same syllabic position, the constraint was obeyed, otherwise it was violated. When analysing the errors for evidence of the syllabic position constraint, syllable boundaries were placed in such a way as to maximise the onset of each

syllable (Kaye, 1989; Levelt, 1989). That is when syllable boundaries were determined, each consonant preferentially formed a unit with the next vowel, rather than the previous one. For example the word "evidence'" would be syllabified as *e.vi.dence*, "tomato" as *to.ma.to,* and so forth. The possibility of a phoneme participating in more than one syllable (i.e. ambisyllabic phonemes) as in "retinal" -> *re.tin.nal* was not addressed when analysing the data. The results are presented in Table 2.3.

Table 2.3.

Percentage of errors obeying syllabic position constraint.

error type		obeyed	violated
within-word exchanges	C (N=50)	80	20
	V (N=1)	100	0
between-word exchanges	C (N=173)	99.4	0.6
	V (N=17)	94	6
within-word and between-word exchanges	C (N=223)	95	5
	V (N=18)	94.4	5.6
migrations	C (N=25)	100	0
ambiguous exchanges/migrations	C (N=30)	100	0
anticipations	C (N=733)	91.7	8.3
	V (N=128)	69.5	30.5
perseverations	C (N=534)	88.4	11.6
	V (N=107)	62.6	37.4

The results show a very strong syllabic position constraint for all types of errors, the effect being strongest for consonants. A further analysis was carried out to see which positions in the syllabic structure participated most often in errors. The results

of the analyses of syllabic constituents (i.e. in terms of onset, nucleus and coda) are presented in Table 2.4. In order that the exchange error analyses could be compared to the other error types, the first position of the erroneous segment in an exchange was referred to as the locus of the error and the latter position the source. This is made clear with an example as in [2.23].

[2.23] portar and **m**estle -> mortar and pestle

In this case the locus is the /p/ in "**p**ortar" and the source is the /p/ from "**p**estle".

Table 2.4.

Percentage of source and locus of consonant and vowel errors in specific syllable positions.

error type	source: / locus:	coda / onset	onset / coda	onset / onset	nucleus / nucleus	coda / coda
within-word exchanges	(N=51)	17.6	2	78.4	2	0
between-word exchanges	(N=190)	0	0.5	85.3	8.9	5.3
within-word and between-word exchanges	(N=241)	3.7	0.8	83.8	7.5	4.1
migrations	(N=25)	0	0	84		16
ambiguous migration/exchanges	(N=30)	0	0	100		0
anticipations	(N=861)	5.4	1.5	68.4	14.8	9.6
perseveration	(N=641)	6.2	3.4	58.3	16.7	15.3

All of the syllabic position violations were accounted for by consonants in either onset or coda interchanging as the source or locus, with one exception where the coda was substituted by the anticipation of a nucleus (0.1% of anticipation errors) as in error [2.24].

[2.24] I mi**o** vote Conservative -> I might vote Conservative

Nearly all of the violations of syllabic position in the exchange errors were accounted for by the within-word exchanges. Of these errors 40% resulted in a word (as in [2.25]) and 60% in non-words (as in [2.26]).

[2.25] steak -> skate

[2.26] tup -> put

Only one violation of syllabic position occurred in the between-word exchange errors, as in [2.27] (syllable boundaries are indicated at the position of each period).

[2.27] .charse .par.ter. -> .chart .par.ser.

The data for within-word and between-word consonant exchanges where syllabic position is preserved is presented in Table 2.5 along with the chance frequencies. The chance frequency of errors occurring in syllable onsets was calculated by summing the number of syllable onsets in each of the data sets. The chance frequency of errors occurring in coda position was similarly calculated.

As can be seen from Table 2.5, both within-word and between-word exchange errors appear in the onset of a syllable more often than would be expected by chance and less often than chance in the coda of syllables. This was significant in a chi-squared test (χ^2=15.58, df=1, p<0.0001), suggesting that there is a syllable-initial effect involved in consonant exchange errors.

All consonant errors were also analysed to see what proportion of syllable-initial errors were also word-initial. This was to see whether or not the effect could be attributed to a higher than chance probability of word-initial phonemes, rather than just syllable-initial phonemes. Chance frequencies were calculated for within-word and between-word exchanges, assuming that onset phonemes are exchanged randomly.

Table 2.5.

Syllable positions (as frequencies) of consonant exchange errors observing the syllabic position constraint.

error type	onset	coda
within-word	40	0
	134	56
between-word	162	10
	536	332

The probability of a phoneme occurring in a word-initial position can be calculated using Bayes' theorem, in (2.1).

$$P(WI \cap SI) = P(WI|SI) \times P(SI) \tag{2.1}$$

P(WI∩SI) is the probability of an error phoneme being word-initial and syllable-initial, P(WI|SI) is the probability of an error segment being word-initial given that it is syllable-initial and P(SI) is the probability of an error being syllable-initial. P(SI) has been calculated already (in Table 2.5, 0.705 for within-word consonant exchanges and 0.618 for between-word consonant exchanges). For within-word exchange errors, the probability of an error being word-initial given that it is syllable-initial is calculated by dividing the total number of possible within-word exchange errors that involve a word-initial segment by the total number of possible random exchanges, summed in both cases for words of all lengths. This is expressed mathematically according to (2.2) and (2.3), where n is the maximum word length (in syllables), l_i is the length of a word (in syllables), $F(l_i)$ is the frequency of words i syllables long and R_i is the total number of possible random exchanges between syllables in a word i syllables long.

$$P(WI|SI) = \frac{\sum_{i=2}^{n}(l_i - 1) \times F(l_i)}{\sum_{i=2}^{n} R_i \times F(l_i)} \tag{2.2}$$

$$\begin{aligned} R_i &= l_i - 1 + R_{(i-1)} \\ R_1 &= 0 \end{aligned} \tag{2.3}$$

For between-word exchanges, it was not feasible to calculate chance for every word in the data, and instead average values for l_i and the number of words between the source and the locus (inclusive) were used. First, the average values were calculated and then the chance frequency calculated based on the average values. The number of words between the source and the locus (inclusive) of each error, and the number of syllables in each of those words were counted and the average values were then obtained. The results are shown in Figure 2.13 and Table 2.6.

As can be seen from Figure 2.13, the distribution of both the length of words involved in errors and the number of words between the source and locus of each error is skewed. The average form of a between-word exchange is of the same form as the error in [2.28], i.e. between two adjacent words each containing one syllable.

[2.28] **n**ate **l**ight -> late night

Table 2.6.

Statistical summary of word length and error span of between-word exchange errors.

		mean	mode	median
word length (in syllables)	(N=411)	1.38	1	1
number of words between source and locus of error	(N=173)	2.43	2	2

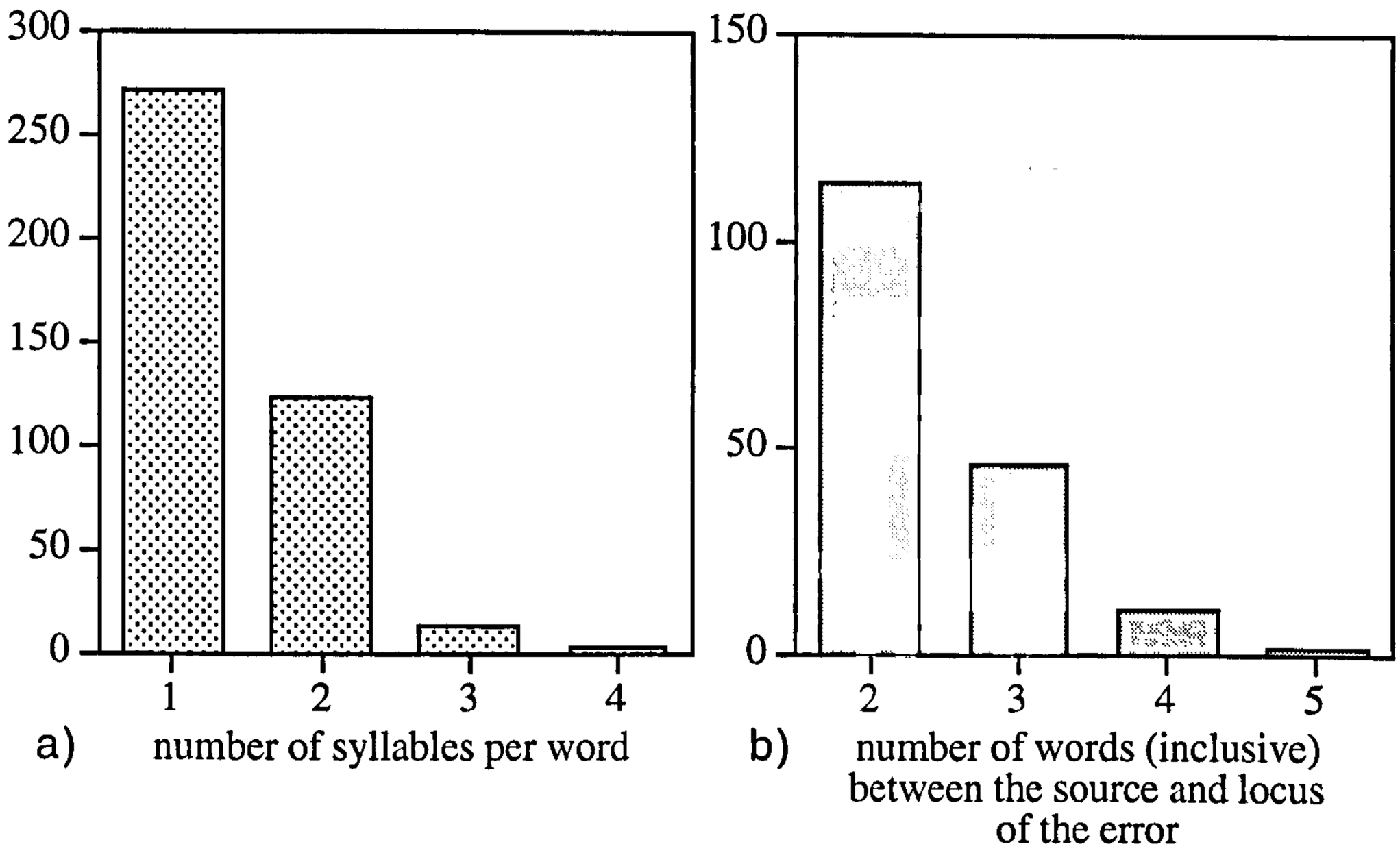

Figure 2.13. a). The distribution of l_i (word length, in syllables) of all words participating in between-word exchange errors in the corpus. b). The distribution of the number of words between the source and locus of each error (inclusive), for the same errors counted in a).

The average form of the error was used as the basis for calculating the chance probability P(WI|SI) for between-word exchange errors, that is the chance that an error segment is word-initial given that it is syllable-initial. On this assumption, for between-word exchange errors, P(WI|SI) = 1. Substituting back into Bayes' theorem in equation (2.1), P(WI∩SI) can now be calculated for both within-word and between-word consonant exchange errors. The results for chance, with actual data are presented in Table 2.7, for word-initial error segments occurring as the locus and source of errors.

Table 2.7.

Percentage of locus and source of word-initial consonant errors.

		onset locus			onset source		
		word initial	not word initial	not onset	word initial	not word initial	not onset
within-word exchanges	C (N=50)	32	66	2	0	82	18
	chance (N=190)	37.4	33.1	29.5	0	70.5	29.5
between-word exchanges	C (N=173)	87.9	5.7	6.4	88.4	5.8	5.8
	chance (N=868)	61.8	0	38.2	61.8	0	38.2
within and between-word exchanges	C (N=223)	75.8	18.8	5.4	68.6	22.9	8.5
anticipations	C (N=733)	54	28.2	17.7	53.1	33.7	13.2
perseveration	C (N=534)	40.3	33.9	25.8	40.8	36.7	22.5

The results in Table 2.6 show that onset exchanges in within-word errors involve word-initial segments less often than would be expected by chance and non-word-initial segments more often than would be expected by chance. However, the same effect is not seen for the between-word exchanges, in that syllable-initial errors which are also word-initial occur more often than would be expected than chance.

In summary, the evidence from the within-word exchange errors suggests a strong syllable-initial effect but no word-initial effect, whereas the evidence from between-word exchanges supports both a syllable-initial and a word-initial effect.

2.2.3.4 Phoneme Repetition

The immediate environment of phonemes involved in errors was analysed to see whether or not the phonemes surrounding them could play a role in the mechanism that produces such errors. This has been defined earlier as the environment constraint. The data are presented in Table 2.8 in two ways. The first column of figures in the table shows the proportion of errors in which the phoneme that either preceded or followed the source and loci segments was the same. For example, in [2.29], the source phoneme, /s/ in "Sunday", and the locus phoneme /s/ in "sudge", are both followed by the repeated phoneme /^/ (as italicised *u* in [2.29]). In [2.30] the source phoneme /n/ in "rained", and the locus phoneme /n/ in "ganes", are both preceded by the repeated phoneme /eI/ (italicised *a* and *ai* in [2.30[).

[2.29] s*u*dge f*u*nday -> fudge Sunday

[2.30] g*a*nes r*ai*med off -> games rained off

The first two columns in the right-hand side of the table show what proportion of repetitions preceded (proactive repetition, as in [2.29]) versus followed (retroactive repetition, as in [2.30]) both error segments. Finally the last column shows what proportion of repetition errors were of the form as in [2.31], where the phonemes both before and after the error segments are the same.

[2.31] my *n*ew*t*s on *No*t*o*n -> my notes on Newton

For these analyses, the chance frequencies of repeated adjacent phonemes is not readily apparent. Although the observed distribution of repeated phoneme errors cannot be compared to the chance frequency of occurrence, certain patterns can be observed. For most of the migrations, anticipations, perseverations and consonant exchange errors, there was no repeated phoneme involved in the errors. Within these error types vowel errors showed a slightly greater tendency to involve repeated phonemes, although the number of vowel errors is much smaller than consonant

errors so marginal differences between them may not be interpretable as such. The reverse scenario for vowels was observed for exchange errors in that twice as many vowel exchange errors involved a repeated phoneme than did consonant exchanges. Again, the number of between-word vowel exchanges is quite small and the data may not be representative of a general trend.

Table 2.8.

Proportions of errors (in percent) involving repeated versus non-repeated phonemes and proactive versus retroactive repetitions.

error type		repetition			locus of repeating phoneme		
		before or after	none		before only	after only	both before & after
within-word exchanges	C (N=50)	26	74	(N=13)	38.5	61.5	0
	V (N=1)	0	100		0	0	0
between-word exchanges	C (N=173)	28.3	71.7	(N=49)	14.3	81.6	4.1
	V (N=17)	70.6	29.4	(N=12)	50	33.3	16.7
within and between-word exchanges	C (N=223)	27.8	72.2	(N=62)	19.4	77.4	3.2
	V (N=18)	66.7	33.3	(N=12)	50	33.3	16.7
migrations	C (N=25)	20	80	(N=5)	0	100	0
ambiguous exchange/ migrations	C (N=30)	100	0	(N=30)	80	0	20
anticipations	C (N=733)	29.3	70.7	(N=215)	51.2	38.6	10.2
	V (N=128)	42.2	57.8	(N=54)	35.2	48.1	16.7
perseverations	C (N=534)	30.7	69.3	(N=164)	55.5	36	8.5
	V (N=107)	34.6	65.4	(N=37)	27	64.9	8.1

A different pattern can be seen when the exact position of the repeated phoneme is taken into account. Consonant anticipations, perseverations and between-word vowel exchanges show proportionally more proactive than retroactive repetitions around the error segments. On the other hand, vowel anticipations, perseverations and consonant exchanges have proportionally more retroactive as opposed to proactive repetitions.

2.2.3.5 Lexical stress similarity

The only errors where an unambiguous intonation contour could be defined were the within-word exchange errors because lexical stress was not confounded with sentential stress. It was found that 60% of within-word consonant exchanges (N=50) occurred between syllables of different lexical stress and 40% occurred between equally stressed syllables.

2.2.4 Summary of Results

Error segments are more likely to be displaced or intrude as an additional phoneme when the source segment is nearby in the utterance. The range of influence is different across error types; the smallest range is observed for exchange errors, then anticipations and finally the longest range of influence is seen for perseveration errors. Although the range of influence is different, a similar relationship between the distance between the source and locus of errors holds across all error types, with most errors occurring over a range of 3 syllables. The proximity analyses yielded results very similar to those of MacKay (1970).

Similarity tendencies appear to be present in more than one dimension; phonetic and syllabic position similarity both play a role in speech errors. Errors are more likely to involve segments that share many articulatory features and occupy the same

syllabic position. Segments at the beginning of syllables and at the beginning of words seem particularly liable to play a role in speech errors.

The contextual similarity, or the environment constraint on the errors is also reflected in the results, although there is no general pattern for all types of errors. Consonant anticipations and perseverations and vowel exchange errors showed a higher frequency of proactive phoneme repetition, but vowel anticipations and perseverations and consonant exchange errors showed a higher frequency of retroactive repetition. The pattern of the environment constraint is different for vowels and consonants within each error type.

Of the errors analysed in terms of lexical stress, the results did not show (as has been seen in other corpora, e.g. Boomer & Laver, 1968) a great tendency for exchanging phonemes to have the same lexical stress values, neither did they show a bias for segments in stressed syllables to act predominantly as source or locus, or vice versa. Finally, all the errors analysed were pronounceable sequences of phonemes.

2.3 Conclusion

The pattern of speech error data found in exchange, migration, anticipation and perseveration errors in Harley and MacAndrew's (1995) corpus support the hypothesis that movement errors are facilitated by the similarity, in several respects, of the segments involved in the error. The temporal similarity (i.e. the distance constraint), syllable position constraint and phonetic similarity are particularly strong factors.

A particularly salient factor is the distance constraint that seems to be seen for all types of error in the analyses. Segments involved in errors are more likely to occur in adjacent syllables rather than several syllables apart and proportionally less errors occur as the distant between erroneous segments increases. The proximity

functions are similar for exchange, anticipation and perseveration errors, suggesting that segments are similarly activated both proactively and retroactively during the production process. Hence a model of speech production that is capable of accounting for these patterns of speech error data must not only allow the simultaneous activation of phonemes before and after they are uttered, but must also capture similarity along more than one dimension, and at least along a temporal dimension.

In the next chapter, I describe some specific models of speech production and how they bear on the movement errors found in the data. The main aim is to ascertain how well these models can account for the pattern and nature of sound movement errors from Harley and MacAndrew's (1995) corpus.

Chapter 3

3. Models of Speech Production

In this chapter I begin with a description of a general model of sentence processing, using Garrett's (1975) model. Other models that are consistent with Garrett's general assumptions about the mechanics of serial processing are presented (e.g. Lapointe & Dell, 1989; Shattuck Hufnagel, 1979) and the general paradigm of serial processing models is discussed in light of its account of the serial order problem observed in human speech errors. In reviewing these models, specific interest is paid to the stage of phonological processing when the acoustic signal is ordered and articulated. Finally, the chapter concludes with a summary of the serial processing theories reviewed, their limitations (specifically related to serial order of speech sounds), and a statement of the need to look for alternative theories, characterised by a more implementational approach.

Early theories of sentence production developed from error analyses, drawing on the patterns and constraints exhibited by speech errors. Garrett (1975) focused on the representation and planning of speech in his model of sentence production, which he supported with analyses from his collection of speech errors. He observed that speech errors were constrained to the same unit, that is words exchanged with words and phonemes exchanged with phonemes, but the two units did not exchange with each other. From this he developed a model that consisted of discrete processing levels. The representations on which each level could operate were restricted to those which were represented at that level and processing occurred in a strictly sequential

manner from one stage to the next. The process of lexicalisation in Garrett's model is a two-stage independent levels model, although there is some controversy over the number of stages involved in lexicalisation, where evidence for one-stage models is also found in the speech error data. I shall return to this after a detailed description of Garrett's model. Garrett's (1975) model focused on syntactic formulation. However, the emphasis in this chapter is on phonological encoding because this is the stage where the serial order problem arises. The rest of the chapter therefore concentrates on theories of phonological encoding, including frame and slot and chaining theories, and examines how well they are able to account for the data as analysed in chapter 2.

3.1. Garrett's (1975) Model

Garrett's early model of speech production conformed to the ideology of serial processing. It proposed that speech is initiated with the formulation of an intended message which is processed in stages to produce a stream of phonemes. This type of model has been characterised as a top-down serial processing model (Harley, 1984). There are four important features of Garrett's model that define it as such. Firstly, Garrett proposes speech production occurs via a four level representational system consisting of a message level, a dual-syntactic level (comprising a functional level and a positional level) and a sound level of representation. Secondly, these levels are assumed to be independent of each other and the processes responsible for producing each level of representation are strictly sequential. That is, each process can only proceed after preceding processes have terminated. This type of processing, where activation across levels does not coincide has also been referred to as a discrete stages model (Levelt et al., 1991). Lastly, Garrett distinguishes between *function* words (words that express grammatical properties) and other grammatical elements, and *content* words (words that express conceptual meaning). He proposes that content

words are stored separately from function words, and are accessed at different times. The model is summarised in Figure 3.1.

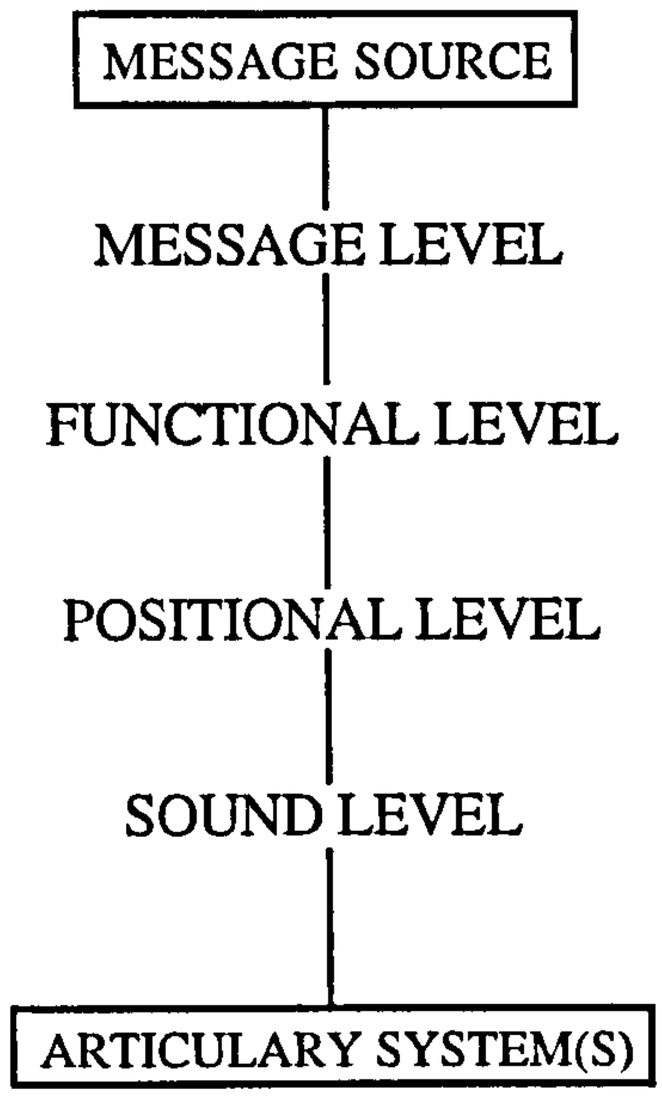

Figure 3.1. Garrett's serial model of speech production. The dual-syntactic level consists of the functional and position level.

The first stage of the process starts with activation at the message level. This activity is an abstract representation of the overall meaning of what is to be said. For example the message may be to convey that Fred had washed the windows. The next stage gives rise to the functional level, creating a grammatical outline for the utterance. The grammatical outline is structured according to the syntax of the language, in terms of the form class of its constituents, such as subject, object, determiners, adjectives, tense and so on. The functional relationships between the constituents are formulated at this stage, but no ordering information has yet been specified. The meanings of the concepts to be expressed are now available, but the particular words are not yet chosen so there is no phonological information available and hence no phonological activity, and lexical entities are accessed by their semantic and syntactic content. The output from this stage is known as the functional level, which in essence specifies the grammatical relationships between elements of the

sentence. For the example message given above, we get an abstract semantic specification of the message to be expressed. In this case,

SUBJECT = "Fred", VERB = "wash", OBJECT = "window"
TENSE = past, NUMBER of OBJECTS = many

Next, an abstract syntactic frame is specified:

N_1 V[+PAST] (DET) N_2 [+PLURAL]

but no information regarding the serial order of the constituent parts of the sentence is specified yet.

The next stage, producing the positional level, fills and orders the syntactic outline with appropriate words. The syntactic rules of the language operate in parallel with the lexicalisation of the content words to give rise to this serially ordered level. The basic units of meaning, or root morphemes, are accessed from the lexicon using the information from the functional level, and their phonological forms are inserted into the slots of the syntactic frame. The positional level has access to phonological information, but no longer has access to semantic information. This level specifies the final position and ordering of the words in the sentence. Garrett proposes that this stage is a dual stage process. The first stage retrieves the root morphemes and inserts them in the frame, for example,

/Fred/ /wash/ [+PAST] (DET) /window/ [+PLURAL]

while the second stage accesses the phonological representations of the function words and other grammatical elements, such as affixes etc. This gives the form of the sound level (for clarity, phonetic symbols have not been used here),

/Fred/ /washed/ /the/ /windows/.

The final stage of Garrett's model specifies where the utterance is translated into a form suitable for driving the articulatory apparatus, which actually produce the speech sounds of the utterance.

In support of his model, Garrett distinguished between exchange errors (as in [3.1] and [3.2]) and other non-movement, or substitution errors (as in [3.3] and [3.4]).

[3.1] That's what **T**omsky was **ch**alking about ->...Chomsky was talking...

[3.2] If I talk **it** into **him**. -> ...him into it.

[3.3] I just like whipped cream and **mushrooms.** -> ...cream and strawberries.

[3.4] All I want is something for my **shoulders.** -> ...for my elbows.

Garrett found that although both word and sound exchanges tend to occur within a clause, sound exchanges are far more constrained by distance than word exchanges. Sounds only exchange over a small distance, typically constrained to the same phrase, whereas words can exchange over phrases and even over distinct surface clauses. The *form class constraint* describes the observation that words which exchange tend to come from the same syntactic category (i.e. content words exchange with content words and function words exchange with function words, but exchanges between content words and function words are very seldom observed), whereas this does not matter for exchanging sounds. The form class constraint is even more marked (almost to the point of being absolute) for the cases where words exchange from distinct surface clauses. This observation is used to support the separation of the functional and positional level in Garrett's model. The word exchange errors are posited to occur during the processing at the functional level, and the sound exchange errors occur during the processing of the sound level (i.e. when the serial order of the phonological elements are determined). Thus the functional level, which operates on the constituents of a whole phrase or utterance and their relative positions, accounts for the wider range of movement in word exchange errors when the constituents are

incorrectly ordered. This occurs when abstract lexical items are put in the wrong part of the syntactic structure. At the positional level, processing is far more constrained by distance as each morpheme is inserted into the syntactic frame, and hence the range of movement for items incorrectly placed at this level will be far more local. Sound exchange errors occur at this level when the process that inserts the phonological information into the frame malfunctions. These differences in constraints on word versus sound exchanges in Garrett's analysis of speech errors are given as support for his hypothesis that the functional level should be made distinct from the positional level of speech production.

Another hypothesis to emerge from his analysis, aside from the serial ordering aspects of speech, was the distinction between content words (words that convey semantic information) and function words (e.g. words such as "the", "a", "of", which convey grammatical information). His analysis of speech errors revealed that content words only ever exchanged with content words and likewise for function words. Given this, together with the idea that content words are not tied to their syntactic frames (i.e. they are retrieved from the lexicon based on their semantic content, unlike function words or grammatical elements), Garrett argued that content words and function words formed distinct categories. Further evidence for the independent categorisation of content and function words came from the fact that grammatical elements accommodate to their environment when an error occurs. For example, when the words in the target phrase "aunts money" exchange, the error is manifest as [3.5] rather than [3.6].

[3.5] aunts money -> **money/z/ aunt**

[3.6] aunts money -> **money/s/ aunt**

The plural affix appropriate to the exchanged word "money" is /z/, rather than the affix /s/, originally intended for "aunt". The accommodation of the plural affix to the new environment supports the theory that grammatical elements are stored

separately and accessed at different times to content words. It also supports the idea that processing at the positional level involves two stages; the latter stage being the process of inserting the function words and grammatical elements. This occurs after the root morphemes have been selected. Accommodation also supports the distinction of a level of processing for sound planning from a level at which word exchanges occur, because accommodation to the environment can only happen after the positions and form of the root morphemes have been decided, i.e. at the positional level or after. The account given of the various speech errors so far seems to support Garrett's sentence production model and assumption that interacting elements must be from the same level of processing.

Garrett (1976) reinforced his claim that movement errors are constrained by certain syntactic factors which support an independent two stage process of syntactic planning. The finding of semantic or phonological similarity between exchanging elements within the error corpora, would contradict his theory. Although such similarities clearly exist for non-movement errors (e.g. the examples in [3.3] and [3.4] show a semantic relationship between target and intruding words), Garrett found no significant effect of semantic or phonological similarity between either exchanging words or sounds in his original corpora.

Other models also developed from Garrett's model (e.g. Lapointe & Dell, 1989; Levelt, 1989), elaborating on the processes of lexicalisation and phonological encoding.

3.2. Lexicalisation

There are broadly three key issues that are repeatedly raised by research on lexicalisation: the number of stages necessary to retrieve a word, the level of interaction between relevant information and the time course of the whole process.

Fay and Cutler (1977) used evidence from word substitution errors to support a one-stage model of lexicalisation. This occurs when phonological information is derived directly via semantic information. They posited that the lexicon is phonologically arranged and access is achieved via a semantic network, like that in Figure 3.2. Semantic word substitutions, such as "trombone" for "ukulele" occur when an error is made traversing the network. Phonological word substitutions, such as "eucalyptus" for "ukulele" occur when the wrong phonological form is chosen once the end of a branch in the semantic tree has been reached. Fay and Cutler (1977) also argued that the same lexicon was used in both production and comprehension.

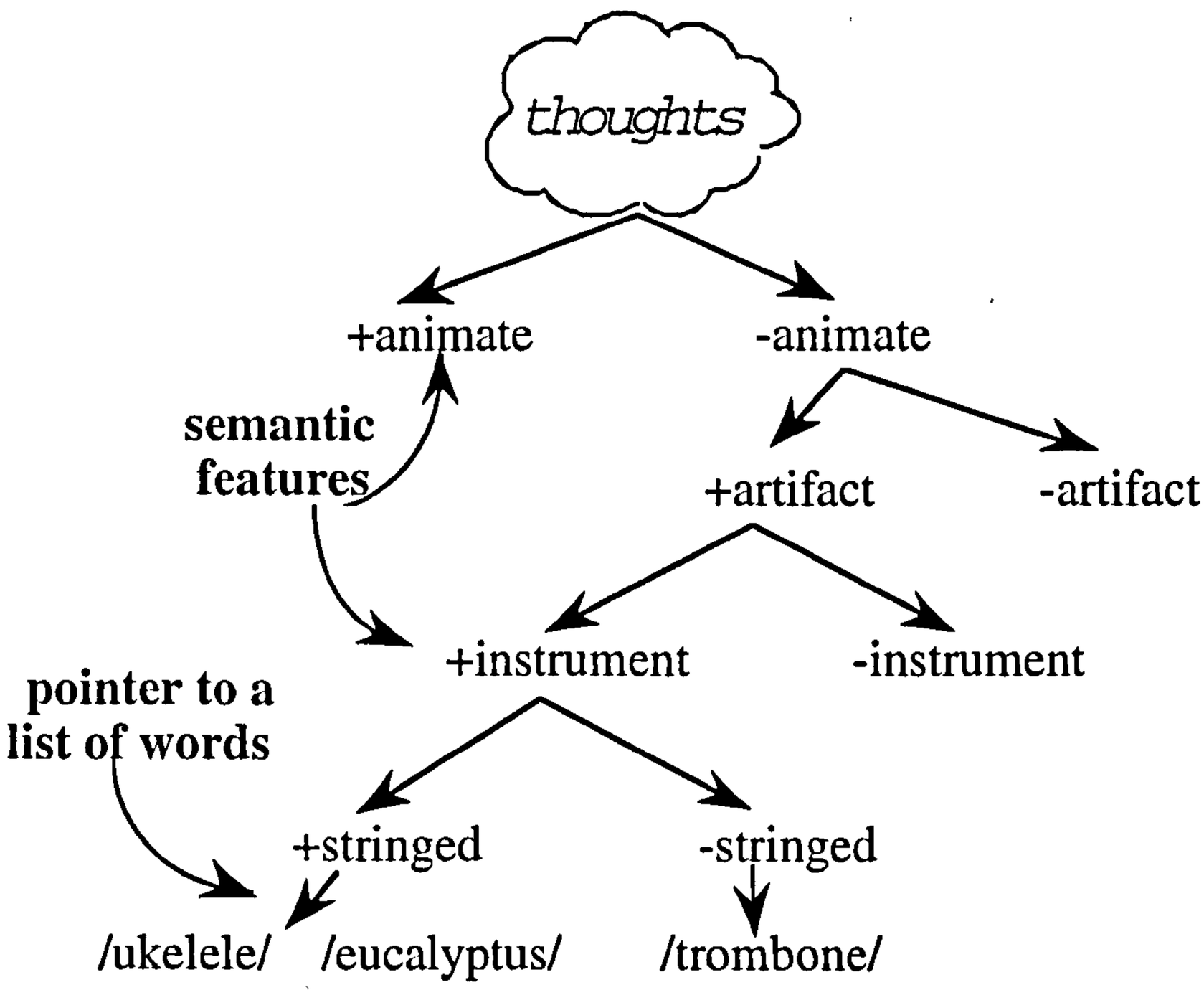

Figure 3.2. Fay and Cutler's (1977) semantic network. Access to the phonological form of words is achieved by traversing the network of semantic markers until the appropriate form is reached.

However, Levelt et al. (1991) argued persuasively that lexicalisation is an independent two-stage process. In their study, subjects were shown pictures of simple objects (e.g. a sheep), which they had to name as quickly as possible. At the same time as they were naming the picture, they also heard a word through a set of

headphones which they had to respond to as fast as possible. They were required to indicate whther or not the word they heard was a real or nonsense word. Their reaction times for the audtory lexical decision task were recorded. If stages of lexicalisation are independent, it is not until after competing lexical items (e.g. "cow", "goat", "horse") have been discarded that the target word becomes phonologically active. However, if the stages are interactive, then phonological activity of words that are phonological neighbours of competing lexical items (e.g. "horn") should also be active. Levelt et al. (1991) found that although pictures of sheep speeded up the response to the word "horse", there was no speed up for the word "horn". They thus concluded that lexicalisation comprises two stages with no interaction between the stages. This is compatible with the serial processing paradigm (e.g. Garrett, 1975; Levelt, 1989).

Independent levels models of lexicalisation do not support parallel or multiple processing. This implies that only one message can be active at a time and that alternative message plans never intrude into the specific sentence plan. Secondly, they do not support interaction between levels. Harley (1984) found evidence of speech errors which are mediated by plan-external thoughts. These errors were termed non-plan-internal errors, and can be differentiated from plan-internal errors which are caused by interfering elements of the intended utterance by the use of context etc. Non-plan-internal errors cannot easily be accounted for by an independent levels model. Furthermore, Harley (1984) found evidence for phonological facilitation, an effect characterised by the increased likelihood of errors occurring if the target and error are phonologically similar. Thus errors that occur at a level higher than the phonological level can be influenced by a phonological similarity between the target and error. What this evidence implies is that if low levels of processing constrain or influence the results of higher levels of processing

then the levels are simply not autonomous. Parallel processing and a spreading activation style lexicon is suggested as part of a more comprehensive model.

3.3. Phonological Encoding

3.3.1 Frames and Slots

Garrett's (1975) model was a formal description of a theory of sentence processing. It outlined the stages of sentence processing from the conception of a message to the final verbalisation. The stage at which sound errors occurred was the sound level, where phonological encoding must take place. This process is undoubtedly central to the production of sound errors and has been explained in many models with the use of phonological frames, with empty slots to hold the selected items.

A frame is a device for sketching out the relationships between pieces of information. Frames also allow abstract units of information to be broken down into constituent parts in a systematic way. They specify a series of slots which may be filled with information of the appropriate type. Ultimately they restrict the composition of the output to specified types of information. For example, a phonological frame may specify the form of the syllable, as in Figure 3.3.

Each slot in the frame may be filled only with information that matches the form of each slot. For example, the first slot may only be filled with segments that are marked as permissible syllable-onset segments, the second slot can only take segments marked as nucleus, and so on. The slots can be filled in a left to right fashion, or in parallel. The order of output information is usually read from left to right from the slot order.

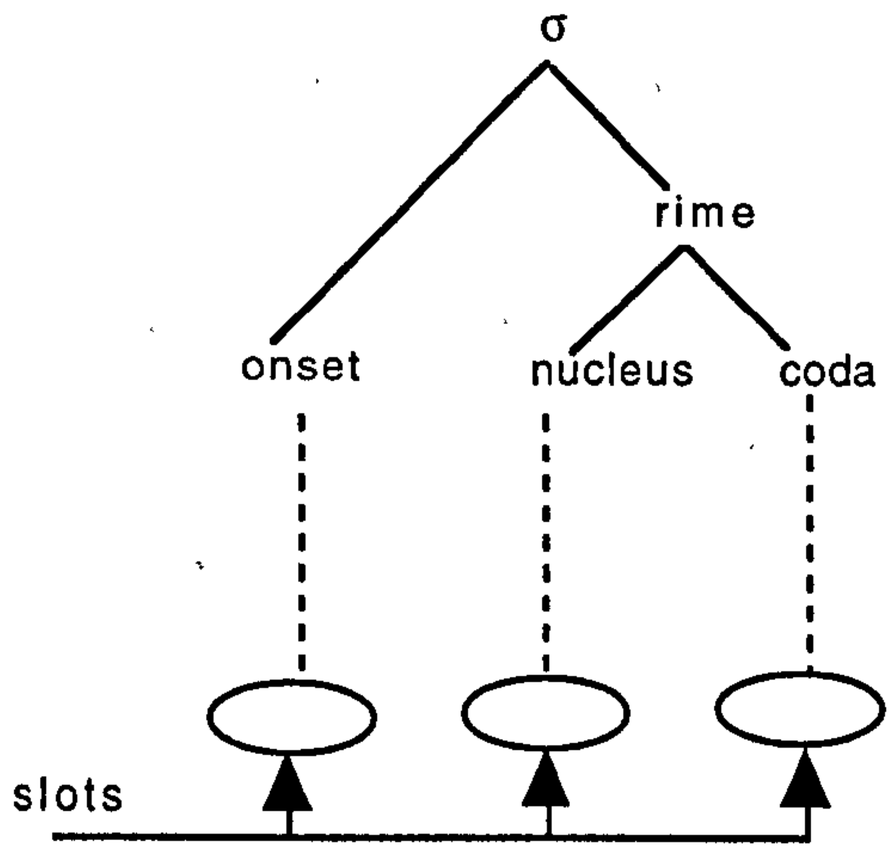

Figure 3.3. An example of a phonological frame and slot mechanism. The frame specifies the formal properties of its constituents and provide slots into which items of the appropriate form may be inserted. σ represents a syllable and the branches represent the constituent parts of a syllable, the onset and rime. The rime also branches into nucleus and coda parts.

Many models of speech production, from both the serial processing paradigm and the interactive activation paradigm assume in one form or another the existence of frames and slots (e.g. Garrett, 1975; Lapointe & Dell, 1989; Hartley & Houghton, 1994).

3.3.1.1 Lapointe and Dell's (1989) Extended Garrett Model

Lapointe and Dell (1989) examined the problem of syntactic frame derivation in detail. They proposed a serial model of speech production based directly on Garrett's (1975) model which they called the "Extended Garrett", or EG model. They filled in the gaps of Garrett's model by making explicit the processes that give rise to each level of representation. Specifically, they focused on the exact nature of the positional frame, and how it is stored and retrieved.

Lapointe and Dell (1989) represented a surface phrase structure fragment with a positional frame. They also differentiated between full phrase fragments that contain slots, markers, inflectional affixes and so on and function fragments that contain only function words, conforming to Garrett's theory that inflectional affixes and function

words are represented separately from content words. A notion store contains a table of the semantic notions associated with function words and inflectional affixes. Each fragment is represented in the fragment store which is linked to the notion store. The connection between the two stores reflects the corresponding semantic notions associated with a particular fragment. A pattern matching algorithm uses the functional level representation to search through the notion tables and access the appropriate fragments. As part of this process, certain rules are applied that numerically index the retrieved fragments. Fragment combination is then a simple process of arranging them into ascending numerical order. Words are therefore serially ordered by the retrieval and combination of fragments which are accessed by the semantic notions found at the functional level. Lexical stem insertion proceeds as follows.

The lexicon is viewed as a connected network of nodes, each representing linguistic units. It is connected such that the semantic and phonological forms of each lexical item are associated with its syntactic function. The nodes are grouped into levels; a lexical concept level (the functional level representation), a level of stems, function words and inflectional affixes and a phonological segment level. The phrase structure segments are linked to the concept nodes such that when a fragment is activated it either activates the appropriate lexical concept node or, if it is a function fragment, the associated function word node. When a lexical concept node becomes activated, activation spreads through the network in both directions. The lexical stem with the highest activation is the 'winner' and is selected for slot insertion. Function word fragments have no slots and selection of the appropriate function word is automatic. Lexical stem selection automatically prompts the retrieval of a phonological frame whose slots are automatically filled with the appropriate phonological segments retrieved from the lexicon. (A full description of the mechanics of spreading activation theory can be found in Stemberger, 1985.)

Lapointe and Dell (1989) explained how selection and phonological speech errors can be accounted for by competing lexical stems becoming more active than target stems and the failure to link the correct segments to their slots in the phonological frame.

Although Lapointe and Dell provided a specific description of how the pattern matching algorithm works to form serially ordered words, the precise method of serially ordering phonological information is not so clear. Shattuck-Hufnagel (1979) made this process explicit in her model of serial order.

3.3.1.2 Shattuck-Hufnagel's (1979) Model

Shattuck-Hufnagel (1979) used serial order speech error data as evidence for a serial ordering mechanism in sentence production. Her model of speech production consisted of three parts. First, she proposed a dual representation of serially ordered slots and independent target speech segments both at the word and sound level. Secondly, a *scan-copier* selects the target segments and copies them into the appropriate slots of a phonological frame. Lastly, a check-off monitor marks or deletes used target segments after they have been copied into their slots and an error monitor finally checks the resulting string of speech units for suspect (erroneous) sequences. Serial ordering occurs at two stages. Firstly the words are arranged into their correct order. Then the scan-copier copies their constituent phonemes into the appropriate slots in left-to-right order as they are retrieved from the lexicon.

The re-ordering of phonemes (when it is usually assumed that lexical items are retrieved with their phonological units already correctly ordered) was justified with evidence from speech errors that involve the misordering of sounds. System malfunctions accounted for errors. There are three primary malfunctions which occur alone or in combination to produce different types of error. These are the misselection of target segments by the scan copier, premature check-off of target segments by the check-off monitor, and delay or failure to check-off target segments by the check-off

monitor. For example, a phoneme exchange such as [3.7] can be explained by the scan-copier misselecting the word-initial phoneme /k/ and selecting /p/ instead, maybe because they are both word initial phonemes and also share some features.

[3.7] **p**utting and **c**asting -> cutting and pasting

The check-off monitor then correctly marks the /p/ segment as used and proceeds with the rest of the utterance. When it reaches the location where /p/ should be copied, it finds that the best possible match for the slot is the /k/ segment and so uses this segment instead. The check-off monitor marks the segment as used and the mechanism operates correctly until the end of the utterance is reached.

3.3.2 Summary of frame and slot models

The models so far have provided a parsimonious account of certain movement error data by reference to frame and slot mechanisms. Thus when segments are misselected, the mechanism that selects and inserts the segments will still choose segments of the appropriate type. Hence constraints such as the syllable position constraint are easily explained. It is not so clear from frame and slot mechanics however, why certain other constraints, such as the effect of featural similarity on exchanging sounds, or distance constraint should exhibit such a strong effect of proximity. The issue of how the frames are filled (i.e. sequentially as in Shattuck-Hufnagel's account, or in parallel as in Lapointe and Dell's model) does not explain why misselection occurs closer together in the utterance rather than far apart. The temporal order of sounds in these models is translated into a purely positional order and hence there is no representation of temporal processing. The models do not reflect the complex temporal nature of speech and therefore have problems accounting for the effects of the distance constraint on movement errors.

3.3.3 Associative Chaining

Frame and slot mechanisms control the order of successive phonemes by the nature of the frame itself. The segments that fill the slots are only related to each other via the frame. An alternative theory of serial order assigned the relationship of serial control to the segments themselves. These theories are known as associative chaining theories.

Associative chaining assumes that processing is uni-directional, usually in a serial manner from left to right. Experimental data described earlier have suggested that phonological encoding, both within-syllable and between-syllables does indeed proceed in a left to right manner (Meyer, 1990, 1991). This has also been supported with evidence from the speech patterns of aphasics (Kohn & Smith, 1995).

The principle of uni-directional associative bonds was formally described by Wickelgren (1969). Serial order was context sensitive because each phoneme (or allophone in Wickelgren's theory) was associated to the phoneme preceding it. The principle of uni-directional bonds in associative chaining is illustrated in Figure 3.4.

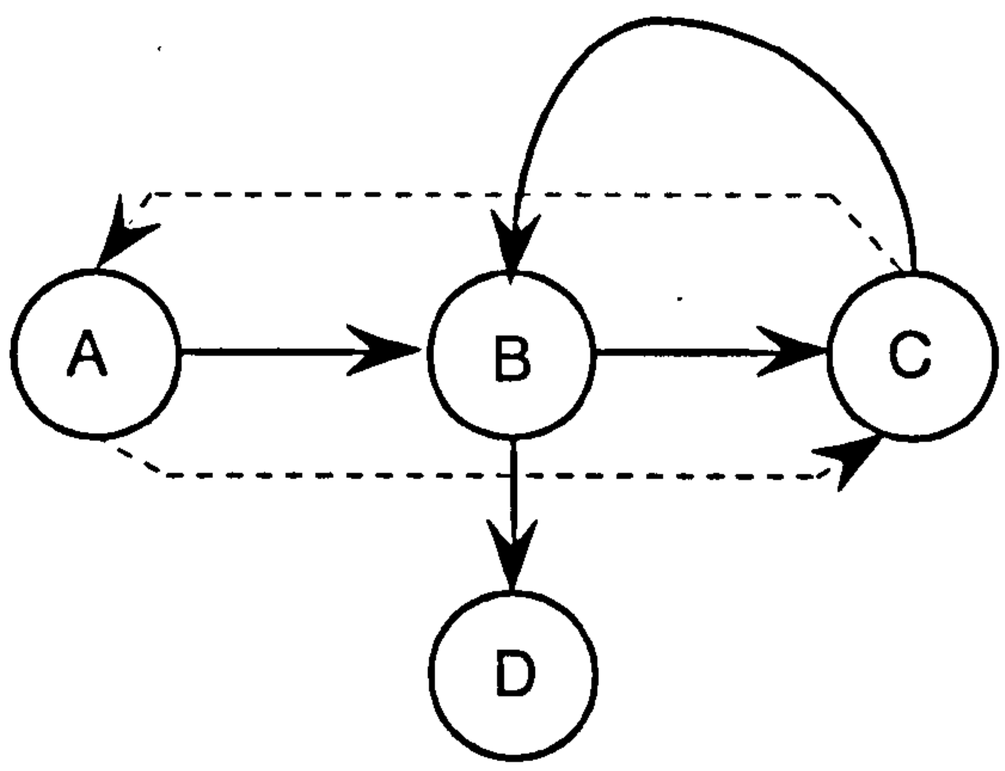

Figure 3.4. The principle of uni-directional associative bonds is illustrated by the solid lines between items A, B, C, D in the diagram. Dashed lines indicate an erroneous placed bond. A permissible sequence may be ABCBCBD.

In Figure 3.4, the sequence ABCBCBD can be produced by traversing the bonds between items.

An important feature of chaining theories is that encoding is achieved in a strict left to right manner and that the associative bonds serve only to activate the next item in the chain. Thus there can be no co-activation of phonemes since once an item has been produced activation is carried forward to the next item which can no longer 'see' the previous one. Associative chaining theories therefore have trouble accounting for certain error types that appear to require the priming of non-adjacent positions both forward and backward, such as exchange errors. Specifically, anticipations, perseverations and exchange errors are not consistent with a 'linear' approach to ordered actions because they require that items are simultaneously represented. A chaining theory would have to explain these movement errors as an error in the associative bonds. For example, if item A is perseverated in the sequence (ABCBCBD), then the erroneous sequence ABCACBD could be produced. This would have to be explained by an erroneous link from C to A, but then another link would also have to find its way back to the original place in the sequence, to C. This hardly seems a well motivated account of such errors.

Chaining is also contradicted by other speech error effects. Chaining theories predict that errors should frequently be preceded by repeated items. For example, in a sequence such as ABCDBF, chaining theory predicts that errors are more likely to occur in recalling C or F because the element that precedes them is the same. However, MacKay (1970) found that repeated phonemes in Meringer and Mayer's (1985) corpus were just as likely to follow errors as precede them. In other words, errors were just as likely to occur in recalling A and D in the example as in recalling C and F. Similar results were obtained from the analyses of Harley and MacAndrew's (1995) corpus, in chapter 2. This is not compatible with chaining theories. Phoneme repetition also facilitates errors in non-adjacent positions to the repeated ones (Dell, 1984). Wickelgren's theory fails to account for such effects found in the data.

3.4. Conclusion

Many models of speech production have been motivated by the patterns found in collections of spontaneously occurring speech errors. Studies of the different units involved in speech errors have provided evidence for the psychological reality of units of speech (Fromkin, 1971) and the levels of processing involved in speech production (Fromkin, 1971; Garrett, 1975, 1976). Many patterns have been observed across different corpora, suggesting that the patterns are more than just an artefact of any one particular collection. These patterns are characterised as constraints on speech in more than one dimension. Structural constraints such as the syllable position constraint and the initialness effect influence which parts of the syllable are most likely to interact in errors. The distance constraint restricts errors to a temporally close range. Phonetic similarities like the environment constraint, the phonotactic constraint, and the feature similarity constraint restrict the nature of likely error segments and the context in which they can occur. Theoretical models have tried to account for these effects, but so far a satisfactory account of especially the distance constraint has not been forthcoming.

Many models of speech production fail to capture the multi-dimensional nature of factors affecting error data as found in the analyses in chapter 2. Exchange errors of single consonants tend to be minimally different in terms of their featural description, which supports the findings of Ellis (1980), MacKay (1970) and Shattuck-Hufnagel and Klatt (1979). Shattuck-Hufnagel and Klatt (1979) suggested that patterns of phoneme substitution errors could be better characterised and accounted for in terms of their featural similarity alone rather than by markedness theories. Markedness theories predict that the strength of phonemes in terms of their frequency or age of acquisition is a determining factor in error data. Markedness theories also fail to account for other similarity effects such as syllabic position of error segments. However, Boomer and Laver (1968) disregarded articulatory

similarity after failing to find any evidence for it in their corpus of errors, and Stemberger (1990) found featural similarity to be a non-significant factor in phonemes adjacent to segments involved in word-initial single consonant exchange errors. Thus the results are incompatible with theories based on the strength or markedness of segments. Wickelgren's (1969) context-sensitive associative chaining theory initially receives some support from the results from the high incidence of proactive repetitions in consonant anticipations, perseverations and between-word vowel exchange errors from Harley and MacAndrew's (1995) corpus. However other error categories (vowel anticipations, perseverations and consonant exchanges) from the corpus show a higher incidence of retroactive repetitions, which present a problem for associative chaining theories because they assume serial order is represented as unidirectional associative links (from left to right) between each phoneme (or allophone). Hence chaining theories predict that errors are more likely to occur if preceded by the same segments rather than followed by the same segments. This is not borne out by the data, where both proactive and retroactive repetition is found. Chaining theories also suffer from the same problem that markedness theories face in that they fail to account for structural constraints which seem to affect errors.

Phonological frame and slot based theories account for the existence of movement errors by the misselection of segments resulting in the insertion of the wrong segment in the frame. Thus structural effects are easily explained in a frame-based account because the existence of the frame ensures only correct types of information will be inserted into the slots on the basis of some type matching process. Associative chaining models cannot account for these constraints easily because they have no representation of structural type information. In models that assume each segment is specified in terms of its phonetic properties, featural similarities are accounted for since similarly represented segments are more likely to be confused. However, both frame-based accounts and associative chaining theories both fail to

account for the distance constraint seen in sound errors. Both approaches fail to represent temporal properties in their accounts and thus cannot account for why errors tend to occur close together, and the particular temporal pattern they yield. The temporal nature of speech is instead translated into a positional representation which subsequently fails to capture the subtle temporal mechanisms that are at issue.

It seems that a more promising approach would be to look to temporal processing mechanisms more directly. One such way of pursuing this approach would be in the paradigm of computational modelling. An implementational approach to modelling temporal processing in speech production has the advantage that it can be easily tested. It is also apparent that a testable model is capable of producing quantifiable results. It seems clear in the case especially of the distance constraint on movement errors that this would be a desirable asset for any theory proposing to account for these data.

Chapter 4

4. Temporal Processing in Computational Models

In chapter 3 I reviewed some models of speech production that provided rather unsatisfying accounts of the temporal processing so clearly characteristic of speaking. Most of these models addressed the temporal problem of serial order by assuming processing occurs in a sequential fashion, structured either by frames and slots, or unidirectional associative bonds between successive elements. In this chapter I address the issue of temporal processing in speech production and how it has been implemented (or sometimes translated into a non-temporal, or spatial, issue) in computational models by different researchers. This is addressed by a computational approach to speech production models, specifically those referred to as *connectionist* models.

4.1 Connectionist Models

From the evidence (e.g. Bock, 1987; Butterworth, 1982; Dell & Reich, 1981; Harley, 1984; Stemberger, 1985) it became apparent that top-down serial processing models of speech production were inadequate to account for certain speech error data and an interactive style of processing was instead suggested. Models developed that allowed the interaction and simultaneous processing of levels, such as the Interactive Activation Model (McClelland & Rumelhart, 1981; Stemberger, 1985).

4.1.1 Interactive Activation Models

Information in an Interactive Activation Model is typically grouped into semantic, syntactic, lexical and phonological levels, with bi-directional links between units at each level (Dell, 1986; Harley, 1984; Stemberger, 1985). Processing occurs in parallel, by a spreading activation mechanism (McClelland & Rumelhart, 1981). How then is serial order achieved? Lexical serial order is achieved by a hierarchical phrase structure frame which directly determines serial order at the lexical level by the differential activation of daughter nodes. This differential activation percolates through the model ensuring the correct serial order of the output right down to the motor-coding level. Lexical selection and sequencing errors occur due to noise either as random variation in the resting levels of units, the frequency of units, or in the spread of activation in the system. During lexicalisation, semantically related words to the target are partially activated. Normally, these are suppressed by the target word, resulting in the correct word being chosen. Sometimes however, if a semantic relative is so active that the target cannot suppress it, it will be chosen instead and a semantic word substitution occurs. Similarly, phonological word substitutions occur when partially activated phonological relatives become over activated such that the target word cannot suppress them. At the phonological level of processing, Stemberger (1982) assumes the lexicon contains a sequence of segments (as indivisible units) for each entry and feature information is contained elsewhere, in a segmental lexicon. Segment selection and word selection is assumed to occur as a parallel process, that is by the activation of elements. Segment substitution errors are explained in a parallel way to those of word substitution errors, i.e. when a relative of the target segment becomes so active that the target cannot suppress it. However, the account of segment sequencing errors is more problematic. Anticipation and perseveration errors occur when first some item information is lost. Next an over-activated segment that is related to the target and that appears elsewhere in the

word is selected because it cannot be suppressed. Why the item information is lost in the first place is unclear, but the real limitation of Stemberger's (1982) model is its inability to provide an account of phonological exchange errors. The same mechanism that accounts for anticipation and perseveration errors cannot be applied to exchange errors, leaving the way in which they occur unclear, as Stemberger (1982) admits, and no alternative account is offered.

Stemberger's (1985) model accounted for the same data that Garrett's (1975) model addressed. Stemberger (1985) was also able to account for data not addressed by Garret (1975). Levels are highly interactive and processing occurs in parallel, in the preferred style of the lexicon. However, the account of sound movement errors (phonological sequencing errors in Stemberger's (1982) terminology) is inadequate. Phonological serial order is derived from phonological frames and is completed directly by the activation of the segmental lexicon. There is no temporal element to the ordering process in Stemberger's (1982, 1985) model, and the structure of speech is processed separately from its content.

4.1.2 Feed-Forward Models and Temporal Processes

Another class of connectionist models, known as *feed-forward* or *back-propagation networks*, provided a tool for the development of models in which temporal and 'frameless' processing could be directly implemented in a single mechanism.

Connectionist models have been widely used to simulate many human cognitive processes, their appeal arising from their ability to provide solutions in complex problem domains, to generalise from similar experiences and gracefully degrade under corruption. In fact, they have been shown to be able to capture a wide variety of input-output mappings with great success (Hinton & Shallice, 1991; Plaut & Shallice, 1993; Seidenberg & McClelland, 1989; Stemberger, 1985). Most

feed-forward models assume a network of connected units, usually arranged into layers. Each unit is a simple processing unit, in that it has a variable level of activation. Patterns of activation across many units are used to discriminate between different states and representations. The units are connected by links of different strengths which are modified by error-correcting training algorithms to best represent the training data, or experience to which the system is exposed. Each unit also has a bias term, which can be thought of as an extra connection from a unit whose activation is always one. Activation travels in one direction from the input layer to the output layer via the connections (and any hidden layers). Data are presented to the model on a set of input units, and the network learns, by modifying its connections, to respond with an appropriate output. An example of a simple feed-forward solution to the parity problem (the output must be 1 if an odd number of 1' s is present in the input pattern and 0 otherwise) is given in Figure 4.1.

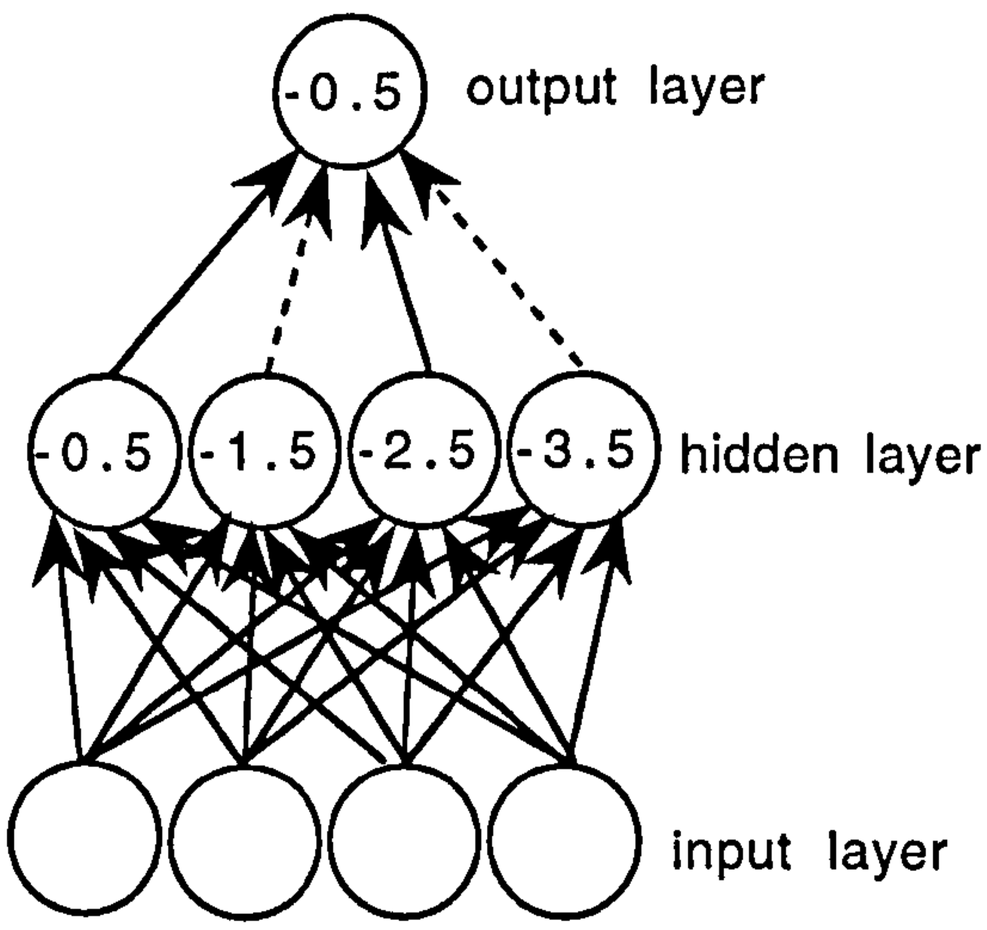

Figure 4.1. A feed-forward network solution to the parity problem. The circles represent units with a bias value indicated by the number inside them. Solid lines indicate connection strengths of +1, dashed lines -1.

The hidden units in Figure 4.1 basically learn to count the number of input units that are on. If an even number of inputs are on, then the net input to the output layer is zero, and vice versa.

There exists a wealth of models of speech production based on the feed-forward model within the literature (e.g. Seidenberg & McClelland, 1989; Sejnowski & Rosenberg, 1987), although very few provide an adequate account of the temporal problem of serial order (but see Houghton, 1990).

When the task involves temporally ordered output the problem becomes more complicated. Serial order and temporal processing is an area that is particularly problematic for feed-forward connectionist models. One obvious approach was to represent *n* states of a sequence by providing *n* sets of units, each to represent a particular state in the sequence. Then position specific units could be used to represent sequential patterns. Three main problems for feed-forward models implemented in this way (originally in Elman, 1990) were described by Chater (1989), and Chater and Conkey (1994). First, past states of the network must be buffered before presentation and remembered so that they can be manipulated in the buffer throughout presentation. Second, the temporal limits of the model must be specified and hardwired a priori. Third, each state of the sequence is completely position specific. This means a sequential pattern such as -ABC- would be very different from (the same) sequential pattern ABC-- (where '-' represents a blank). Various techniques have been employed to try and overcome these problems. For example, Sejnowski and Rosenberg (1987) trained a feed-forward model called NETtalk to perform the appropriate mapping from text to a sequence of sounds. They used a 'moving window' on the input such that the output was a function of the central letter of the window, relative to the letters either side of it. An appropriate sequence of phonemes was produced as the window moved over the input text. However, the size of the window was pre-set and past and future states of the

sequence had to be buffered in some way. Thus a moving window account essentially translated the temporal aspect of the problem into a spatial one. The problems of remembering past states and pre-specifying the extent of the window still remained. These limitations motivated the development of a new type of connectionist network in which temporal processing was implicit, and is described in the next section.

4.1.3 Recurrent Connectionist Models

An inherent feature of connectionist models is their massively parallel nature. That is, they consist of many simple units that interact and are processed simultaneously. Yet for the production of serially ordered sequences, actions must be produced one a time and in the correct order. It is not straightforward to see how parallel systems can be made to produce serially ordered sequences without overcoming the problems described above. The output of feed-forward models is governed solely by the input available to them at the current point in time. That is, they do not have the capacity to remember previous states. When the input-output mappings are temporal in nature, the need for some memory of previous (and later) states becomes apparent. These problems are particularly well illustrated in the domain of speech production. To produce speech, a sequence of phonemes is produced in response to a particular lexical input. For example, the phonemes associated with the word 'cat' are /k/, /@/ and /t/. However, to pronounce the word "cat" correctly, the phonemes must be output in specific order, i.e. first /k/, then /@/ and finally /t/. To produce a sequence of events, it is necessary to remember the previous states in order to determine successive states. Hence, some form of memory of previous states is necessary.

The connectionist enterprise has risen to the challenge of serial order, however, and models have been developed that directly address this issue. For example, Jordan

(1986b) and Elman (1990) have both proposed connectionist architectures suited to producing serially ordered output. Existing connectionist models of serial order which implicitly represent temporal processing can be broadly split into two classes, the recurrent network approach and the graded output activation approach (or competitive queuing). Recurrent networks have been applied to many tasks, especially in the domain of language (Das, Giles & Sun, 1992; Dell, Juliano, & Govindjee, 1993; Elman, 1990; Jordan, 1986b; Plaut & McClelland, 1993; Servan-Schreiber, Cleermans & McClelland, 1991) and are discussed in the next section. I shall defer discussion of competitive queuing models until a much later stage.

4.1.3.1 Jordan's Model

Jordan (1986a) suggested an architecture that is similar to the feed-forward model, but differs by feeding back the output associated with the input on each time-step to predict the next state in the sequence. A time-step is the unit of time dedicated to representing each element of a sequence. Figure 4.2 illustrates the architecture of the Jordan network. The feedback is achieved essentially by copying the activation levels of the output units at time *t*, for use at time *t+1*, to the state units, as depicted in Figure 4.2.

By providing recurrent connections from the output layer of units, an additional set of input units now exist to provide information relevant to past input. At each stage of processing, the model now has access not only to current input, but also an extra 'memory' that retains the history of previous output. Thus the model is able to associate the present sequence output with the current input plus some function of the past states. The architecture places no constraints explicitly on the length of the sequence to be learned, as has been the case in other architectures (Sejnowski & Rosenberg, 1987).

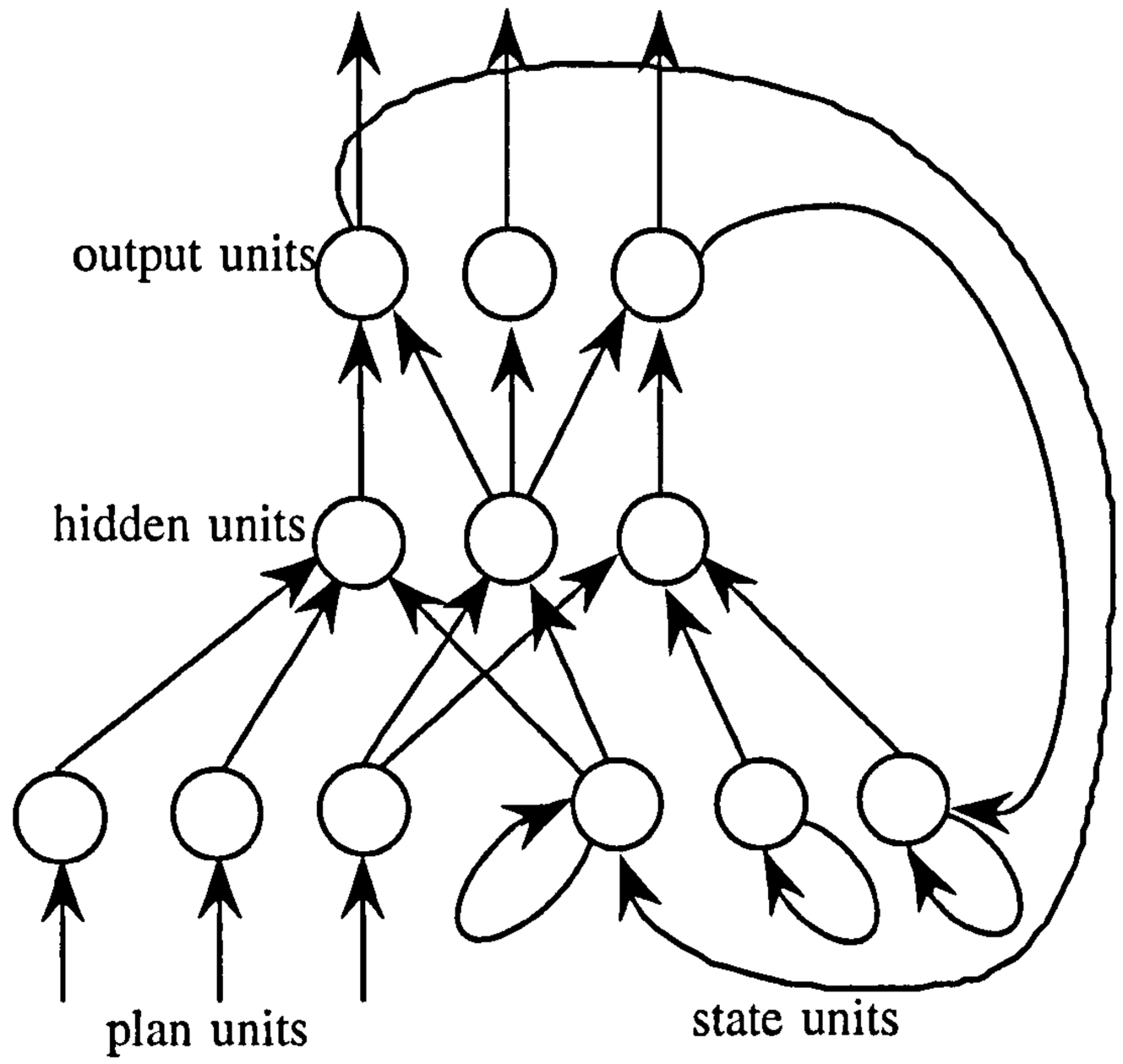

Figure 4.2. The Jordan (1986) Sequential Network.

However, Jordan's sequential network has been criticised on several points. Servan-Schreiber et al. (1991) note its ability to discriminate between different sequences decreases as sequence length increases, while Wang and Arbib (1993) and Dell et al. (1993) point out that sequences with repeated items (e.g. **ABCAD**) or repeated sub-sequences (e.g. **ABCABD**) cause major problems because the repeated constituents cannot be uniquely represented. Jordan overcomes this problem by giving the state units a self-recurrent link, so that their activation at any point in time is a weighted combination of their previous state and the previous output of the network. The network will still fail on repeated sub-sequences above a certain length, however, and the network still lacks the ability to stop producing output when the end of a sequence is reached, and instead cycles continuously.

4.1.3.2 Elman's Model

Elman (1990) proposed a similar approach, based on the idea that the hidden layer of a connectionist model forms an internal representation of the input pattern (Rumelhart, Hinton & Williams, 1986). By feeding back the activation of the hidden layer, the network is able to encode not only the previous state, but (in some sense) all predecessors of the previous state as well. Figure 4.3 illustrates the architecture of Elman's (1990) model. The idea is very similar to that of Jordan, except the hidden layer of units is used to provide the memory for the model rather than the output layer. In this way, the internal representations for the sequence of events are preserved, rather than just the previous raw output. Again, feedback is achieved by copying the activation levels of the hidden units at time *t* for input at time *t+1* to the context units.

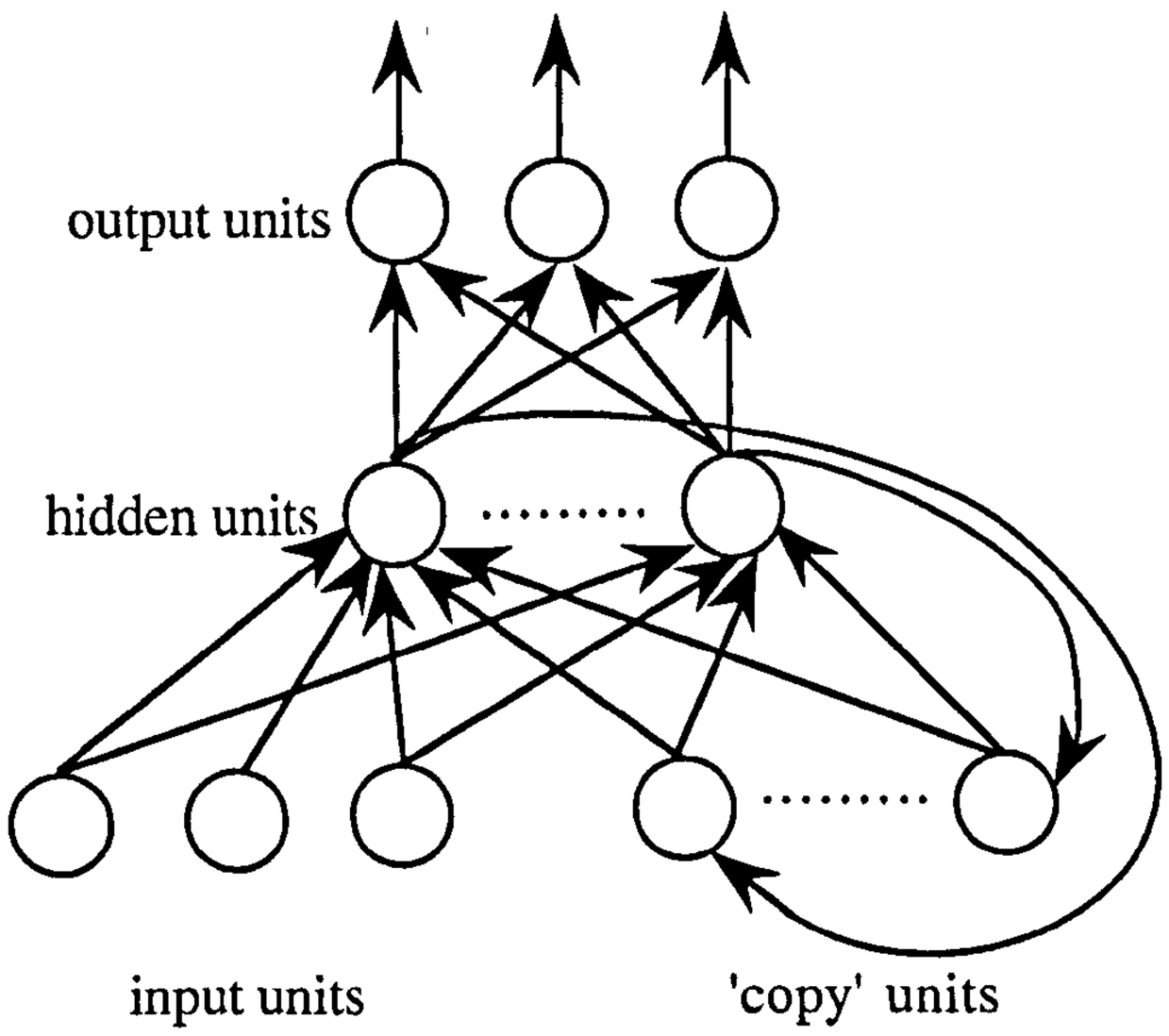

Figure 4.3. The Simple Recurrent Network by Elman (1990).

Using the hidden units as a memory mechanism means that the coded form of the preceding input is remembered throughout the sequence because it is fed back into the network at subsequent time-steps. Remembering the coded input representation as opposed to the single previous raw output enables the architecture to disambiguate repeated constituents and avoid the problem some networks encounter with repetition within a sequence. This is because the coded input representation (found at the hidden layer) is a function of the input and hence different for each sequence (even for the same elements in different sequences). Elman's architecture is well suited to prediction or recognition tasks and has largely been applied in this area rather than production.

4.1.3.3 Training Methods for Recurrent Networks

It may not be immediately obvious what sort of learning algorithm should be used to train a recurrent network. However, this poses no problem as the generalised delta rule also applies to recurrent networks. The generalised delta rule (Rumelhart, Hinton, & Williams, 1986), although originally derived for feed-forward type networks, can be successfully applied to the architecture of the recurrent network. There are two main training regimes which are both based on this learning rule, yet they differ in their implementation.

4.1.3.3.1 'Copy-back' Method

Recurrent networks are implemented with recurrent links from the output units back to the input units. This is achieved by providing the network with an additional set of input units which are copies of either the hidden or output layer at the previous time-step, i.e. as illustrated in Figure 4.2. These copy units (sometimes called 'context' or 'state' units) are linear and have a zero bias to ensure they represent an undistorted replica of the previous activity of the units from which they were copied. Thus the recurrent architecture can be trained by applying the standard

back-propagation algorithm to each time-step in the sequence (see Figure 4.4) and is often termed the 'copy-back' training regime.

For the first step in the training sequence, the context units are set to zero and therefore encode no information of prior learning. The output is computed, based on the first input of the sequence and the error between this output and the desired output is then calculated. The error is propagated through each layer and all weights in the network, including those connecting the context and hidden units are modified accordingly. The activation of the hidden (or output) units is then copied to the context units and coupled with the next input pattern in the sequence provides information about two time-steps in the sequence. Training continues in this way until the end of the sequence is reached.

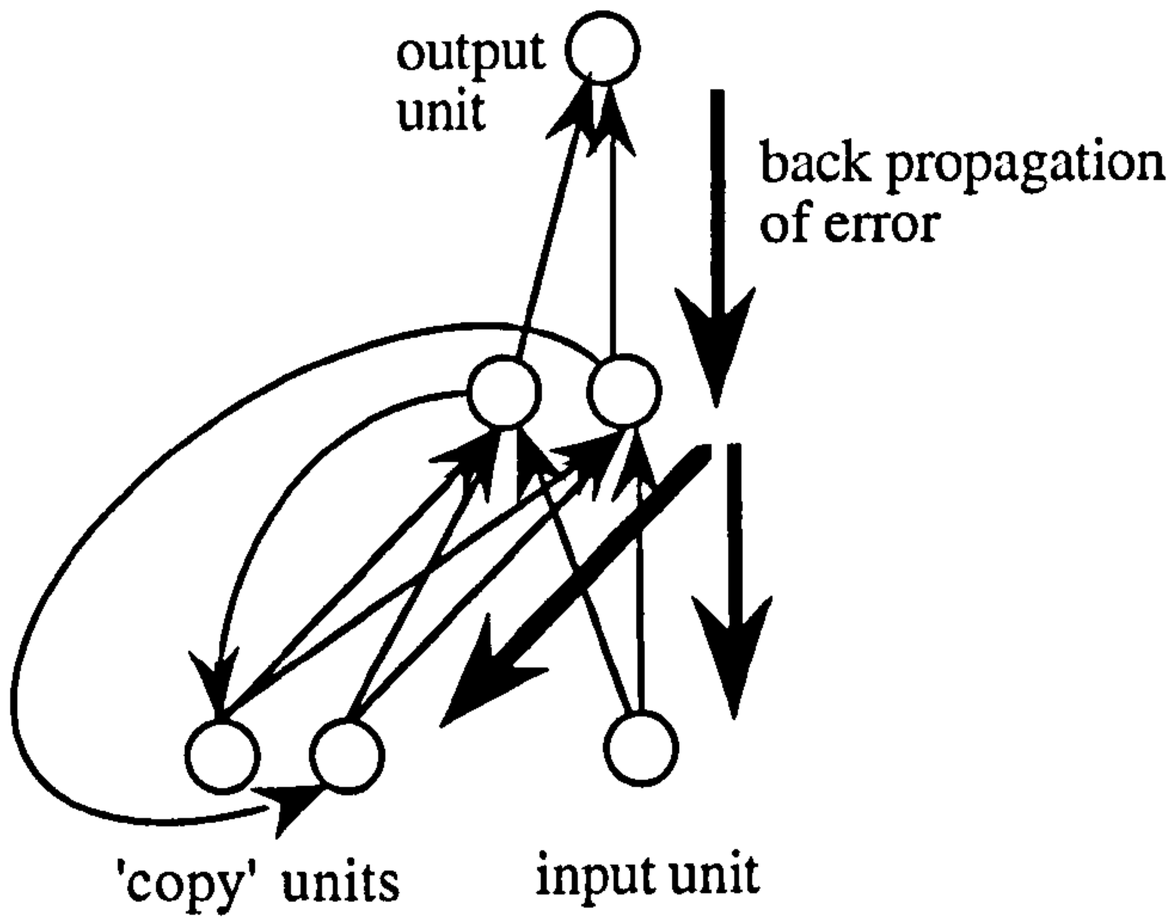

Figure 4.4. The copy-back training regime for a recurrent network, showing the back propagation of the error through the network.

Note that this method uses on-line back-propagation (weights are updated after each presentation of a training pattern), as opposed to the batching method (weight changes are summed for all training patterns in the entire training set before they are implemented).

4.1.3.3.2 'Unfolding' Method

An alternative approach was motivated by the fact that for every recurrent network with feedback connections of one unit time-step, there exists a corresponding feed-forward network (for a finite number of time-steps). The recurrent network can be transformed into a feed-forward network by a process known as 'unfolding' a recurrent network. Unfolding a recurrent network entails duplicating the units in the recurrent network for each time-step in the sequence which yields a tree shaped network with many hidden layers. This unfolded version of the recurrent network can now be trained by the generalised delta rule. This is illustrated in Figure 4.5.

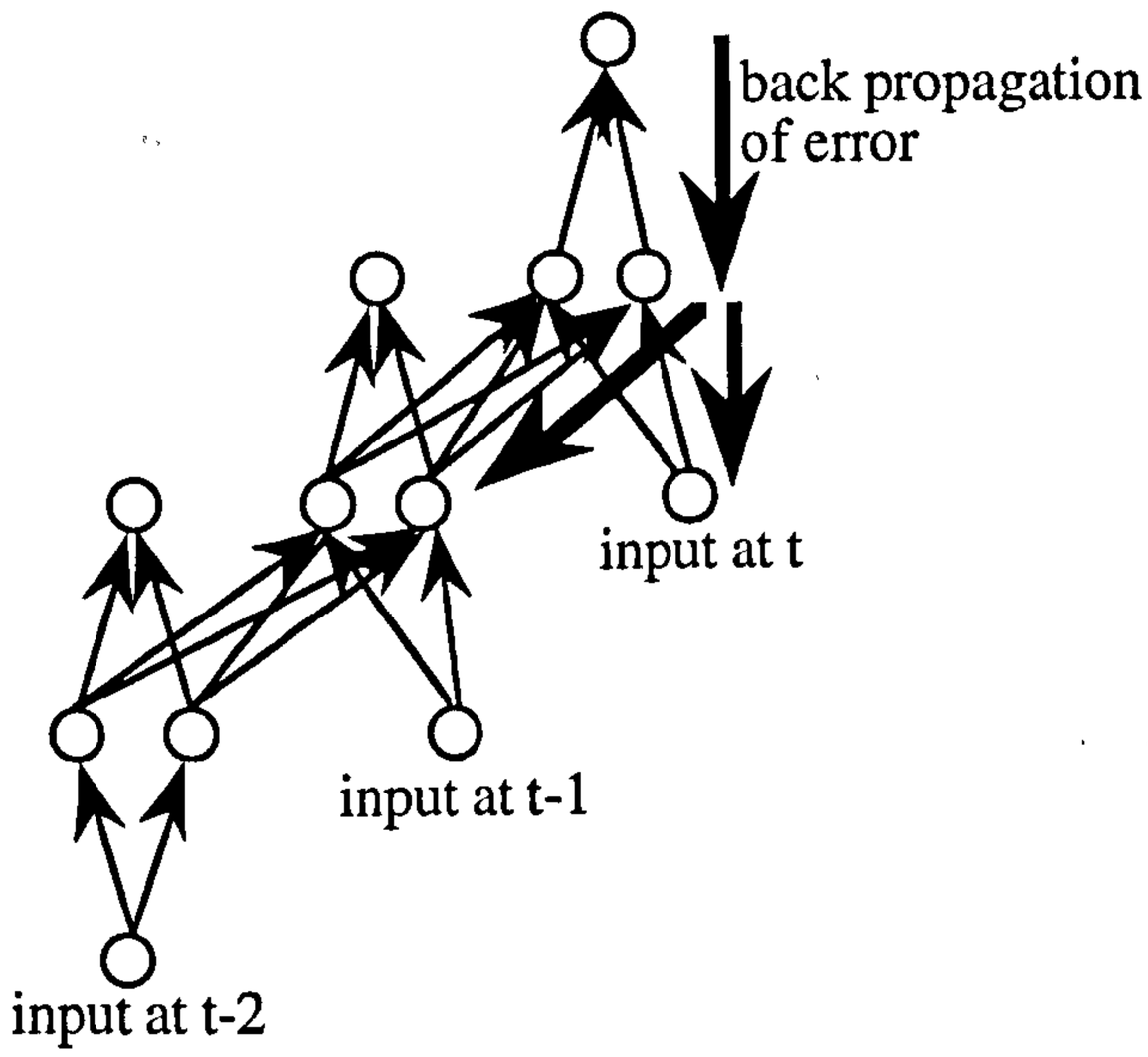

Figure 4.5. Back propagation of error through time in an unfolded recurrent network. Each input represents the relevant input at *t-n* time-steps ago.

The network is unfolded backwards in time, starting from time *t*, so that the error can be propagated back through (in time) through the network. However, the weights on each corresponding link of the duplicate networks must be constrained to be the same. Therefore, the changes prescribed by the learning rule must be the same for each such set. This problem is overcome by summing the weight changes for

each individual link in the set and applying the resulting weight change to each link. Both training regimes (copy-back and back-propagation through time) have their limitations (Chater, 1989).

Chater criticised the copy-back regime on the basis that any encoding of temporal information by the weights happens 'by chance'. He supported this claim with the fact that the copy-back regime takes into account only the most recent weight changes (i.e. one time-step in the sequence), as opposed to the sum of all the weight changes through time (i.e. the whole sequence). If, for any of the steps in the input sequence during training, the sign of these changes differ, then the copy-back regime ceases to minimise the error. However, by computing the weight updates for the entire training set before implementing them (i.e. the batching method, Rumelhart, Hinton & Williams, 1986) this limitation could be overcome. In any case, the parameters of back-propagation can be adjusted by reducing the learning rate so that changes in the weight space occur slowly, rendering the two methods almost identical in practice (Williams & Zipser, 1989). This will prevent the network from settling on a set of weights that although apparently minimise the total error of the network, provide an incorrect solution. However, there is no guarantee that these principles can be in practice applied to all situations or problems. In cases where the copy-back regime fails to satisfactorily learn a data set, Elman (1991) suggests the problem may actually lie in the presentation and nature of the data rather than in the training algorithm itself. Elman (1991) has shown how incremental learning can master complex training sets that have otherwise proved unlearnable by networks.

Back-propagation through time has its limitations as well. Apart from the fact that it is computationally expensive (n number of copies of the unfolded network have to be kept for a sequence of length n), the longer the sequence, the more separated the start of the sequence becomes from the end of the sequence by many hidden layers. Networks with large numbers of hidden layers not only take much longer to train, but

are also less reliable because of the distance through which the error signal must percolate to reach the initial layer in the network. A more worrying drawback imposed by this regime is that the length of the sequence to be learned must be known prior to training, so that the network can be appropriately unfolded. This is because to unfold a network through time, the past states of the network must be remembered until the unfolding takes place, at the end of the sequence.

Recently, Beaufays and Wan (1994) have argued that back-propagation through time and the on-line copy back method reduce to the same algorithm, and that theoretically both implement the same weight updates. Thus there seems to be little difference between the two methods providing that suitable parameter values (particularly for the learning rate) are chosen for each algorithm. However, it does not seem clear that both methods will in practice find the same solution to a particular problem, nor that the two methods are equivalent (in practice) for all problems.

4.1.4 Dell, Juliano and Govindjee's (1993) Model

The models of speech production so far have built their theories on the assumption that the structure of speech is separately represented by syntactic and phonological frames. Recently, Dell, Juliano, and Govindjee (1993) questioned this assumption, suggesting that alternative temporal mechanisms could account for the same speech error data, challenging the necessity of the structure/content distinction made so far. Dell et al.'s (1993) recent model of single word production was based on a recurrent network model. It did not assume any structure in the form of a phonological frame, i.e. no a priori distinction was made between phonological structure and segmental content, but was viewed as an emergent property of the architecture. The model demonstrates how frame-based constraints and behaviour previously assumed to be explicitly rule governed could be accounted for by other means.

Dell et al. (1993) built on the ideas of Jordan (1986b) and Elman (1990) to produce an combination of both architectures. Their model made use of both *internal* (units copied back from the hidden layer, as in an Elman architecture) and *external* (units copied back from the output layer, as in a Jordan architecture) *context* units. They trained their model on two different sets of fifty-word vocabularies, to test the hypothesis that frame constraints on speech errors reflect nothing more than the mass action of experience of vocabulary. They presented the speech error data that has given strongest support to the syntactic frame approach and undermined its necessity by demonstrating that these errors could be predicted and simulated by his model. Constraints that had previously been assumed to be frame-based were explained by effects they termed similarity and sequential bias. The similarity bias effect accounted for why erroneous sounds were likely to be similar to the intended sound because of the similarity between their sets of features. For example, the phoneme /g/ shares more features with the phoneme /k/ than it does with /l/, and hence it is closer (featurely speaking) and therefore more similar to /k/ than /l/. The sequential bias effect accounted for why sound sequences are more influenced by common sequences rather than by sound sequences that have seldom or never been experienced before. These two key effects formed the basis of their account of the frame-based constraints, which are known as the *phonotactic regularity* effect (MacKay, 1970), *consonant-vowel* category effect (MacKay, 1970), *syllabic constituent* effect and *initialness* effect (Broeke & Goldstein, 1980; MacKay, 1970). The phonotactic regularity effect is characterised by the phonotactic regularity of errors, so that the error does not violate the phonotactic constraints of the language. The consonant-vowel category effect reflects the phonological class of interacting segments in that consonants tend to interact with other consonants, and vice versa for vowels. The syllable-constituent effect describes how consonant-vowel clusters are more likely to be involved in errors than vowel-consonant clusters. Finally, the initialness effect is observed when syllable-initial segments are involved in an error.

Dell et al. (1993) explained all of the above phonological frame constraint errors in terms of sequential and similarity bias. They accounted for the *consonant-vowel* category effect in terms of similarity bias, because consonants and vowels are quite dissimilar and hence less likely to replace each other. The *consonant-syllabic* effect was explained by the sequential bias, as vowel-consonant sequences occur within more than one word more often than consonant-vowel sequences, hence vowel-consonant sequences would be more familiar. They also explained the *initialness* effect in terms of sequential bias, because at the beginning of a sequence less information is available as a cue than later on in the sequence when more context is available. Hence errors are more likely to occur at the onset of a sequence. Strictly speaking the model only accounted for a word-onset (rather than syllable-onset) error pattern; and whether there is a real distinction between the two is still unclear. Finally, the *phonotactic regularity* effect was due to a combination of the similarity and sequential bias effects. Phonotactic patterns are sensitive to similarity (hence sequences similar to legal strings are more likely than illegal combinations that are less similar) and the previous context biases the model to produce legal combinations.

The model provides an alternative mechanism responsible for speech error data and therefore an alternative mechanism for producing the serial order of speech sounds. Information is retrieved in serial order from a parallel system, rather than simultaneously as in interactive activation models. There are no frames and explicit rules of frame retrieval mechanisms to impose order on the output; serial order is an implicit, emerging property of the architecture that arises due to the temporal processing embodied in the model.

Although the model is an attractive model of speech production, it is incomplete as a theory of phonological encoding and cannot account for movement errors. It cannot produce movement errors such as exchanges and shifts and any anticipation or perseveration errors it does produce arise as non-contextual substitution errors.

Hence they are not true movement errors. The authors acknowledge that movement errors are problematic within their current implementation. The modifications they suggest to account for exchange errors have not been implemented, so the question remains open as to whether the model would produce exchanges. However, the model does show how a simple mechanism can reveal powerful principles, such as the similarity and sequential bias to account for speech error effects, without recourse to frames and slots. Furthermore, the model also allows a more realistic interpretation of temporal processing to form part of the speech production system.

4.2 Conclusion

The modelling of speech production has moved on from the early top-down serial processing (Garrett, 1975; Shattuck-Hufnagel, 1979) paradigm and phonologically-listed lexicon approach (Fay & Cutler, 1977) to models that view the lexicon as a connected network (e.g. Harley, 1984; Lapointe & Dell, 1989) and models that recognise complete interaction between levels (Stemberger, 1985). However, all of these models, even ones that assume an interactive or parallel processing element (Interactive Activation style models), approach the issue of serial order in the same manner: they all assume an explicit representation of syntactic and phonological frames that must exist a priori. Opinion differs on how these frames are retrieved, either by algorithmic search and rules (Lapointe & Dell, 1989) or spreading activation mechanisms (Stemberger, 1985) to how they are filled, either by repetition of serial events (Shattuck-Hufnagel, 1979) or again by spreading activation mechanisms (Lapointe & Dell, 1989; Stemberger, 1985). All of these approaches fail to recognise in a plausible way the temporal aspect of serial order in speech production.

The recurrent network model, (Elman, 1990; Jordan, 1986) provides a parsimonious account of temporal processing of serial order in a paradigm that is

otherwise a highly parallel system. The temporal aspect of serial order in these models arises in a more plausible manner as a result of the processing of each element of a sequence over time. Recurrent models also have some implementation of temporal memory such that the output at any one time is a function of the current state plus all past states; thus past states during sequence production bear on the production of the current item. The recurrent connectionist approach also has the advantage of allowing models to be developed that need minimal a priori information regarding the process being modelled because they learn from examples, and because they can be implemented they are also more easily tested. Dell et al. (1993) offered an alternative to the explicit representation of serial order by demonstrating how a simple mechanism without rules or frames can produce serially ordered output with their recurrent network model of pronunciation. One crucial limitation of an otherwise successful model of phonological ordering was the omission of any account of movement errors. Exchange errors were specifically problematic in this model because the displacement of one phoneme (by an anticipation) in a sequence would not cause a corresponding tendency for the displaced phoneme to complete the exchange by replacing the source phoneme that had been originally anticipated. Although Dell et al.'s (1993) model has its limitations, it shows how evidence that previously supported frame-based theories of serial order can be accounted for within a recurrent connectionist framework, in which serial order is an implicit property of the model, arising from the parallel nature of simple units. The connectionist models I have described in detail in this chapter can be categorised as recurrent models. This type of model is surely the most promising so far in terms of its ability to reconcile temporal processing and serial order in a parsimonious manner.

Chapter 5

5. Recurrent Network Models of Serial Order

From the literature reviewed in the previous chapter the most viable connectionist approach to the production of serially ordered information so far has been the recurrent network model suggested by Jordan (1986a, 1986b, 1989; Jordan & Rumelhart, 1992). Jordan's aim was to simulate the production of sequences rather than their recognition or classification as other researchers have often sought to achieve (Elman, 1990; Fahlman, 1991; Servan-Schreiber et al., 1991). The task of sequence production is more akin to the human skill of speech production, whereas a sequence recognition or classification system relates more closely to human perception skills such as reading or oral comprehension. Jordan's sequential network (1986b) seems an obvious choice to begin modelling because it is a connectionist model of serial order that has already shown promise in Dell et al.'s (1993) model.

This chapter takes a closer look at the viability of Jordan's network by means of a series of simulations of his model applied to a task designed to provide a more rigorous test of the model's claims to sequence production.

The first in the series is a simulation of Jordan's (1986a) model applied to a sequential learning task as originally documented by Jordan (1986a). This is provided as a replication of Jordan's (1986a) results, obtained from my own program, on which my later simulations of his model are based. Later work by Jordan (1989)

and Jordan and Rumelhart (1992) included more complex versions of his sequential network, but his simplest model is simulated first.

5.1 The Jordan Sequential Network

An exact replication of Jordan's original work was not possible for several reasons. Firstly and most importantly, some pertinent implementation and algorithmic details were not available in his original report. Thus crucial details such as the parameters affecting learning and the exact method of training had not been explicitly stated and could not be replicated with certainty. Parameter values for the learning and momentum rates were not specified, for example. In such situations, I have started by following principles for choosing values from Rumelhart, Hinton and Williams (1986) and varying them as appropriate. Less important differences in model performance will arise owing to technical computational differences between different computers on which the simulations are run, such as the precision of floating point numbers which can vary between different computers. The original work has been followed as closely as possible where the documentation permits, but where ambiguities arose in the documentation of Jordan's model, I have made the most plausible or sensible implementational decisions as necessary. The slight differences in implementation should not affect the overall behaviour of the model. The next section in this chapter reports the best approximation possible of a replication of Jordan's original model.

5.1.1 Model 5.1

A suitable task was chosen from Jordan (1986a), where six sequences were formed by all the possible permutations of the actions A, B and C (e.g. ABC, ACB, BAC, BCA, etc.). Each sequence was distinguished from the rest by a unique plan (input pattern) and each (output) action was represented by a two-bit output vector,

with A = [0 1], B = [1 1] and C = [1 0]. Thus for each sequence, the plan for that sequence would represent the sequential output of the appropriate vectors. For example the plan for the ABC sequence represented the sequential output of actions A[0 1], B[1 1], and then C[1 0]. Two plans were constructed, according to different methods, for each sequence. The methods for determining each plan representation were non-arbitrary and corresponded to the plan representation schemes used by Jordan. The first scheme assigned a pair of units in the plan to represent each action in the sequential order in which it appeared in the corresponding sequence. I shall refer to this scheme as the *serial position* scheme. For example, the plan for sequence ABC was represented by the input pattern [0 1 1 1 1 0]. The first pair of units represent the pattern for action A[0 1], the second represent the pattern for action B[1 1], and the third represent the pattern for action C[1 0]. Thus presentation of the input pattern [0 1 1 1 1 0] represented the sequence of actions ABC. The second scheme assigned one unit in the plan to represent one of the six possible transitions (A-B, A-C, B-A, B-C, C-A, C-B) in the sequence being learned. This will be referred to as the *transitional* scheme. For example, the plan for sequence ABC was represented by the input pattern [1 0 0 1 0 0]. Each bit represents a state transition as just described: the first and fourth bit are on indicating two transitions, the first from A to B, and the next from B to C, giving the sequence ABC.

5.1.1.1 Architecture

The input layer consists of two clusters of units. Six units form one cluster to represent the plan for each sequence to be learnt. These units are set by an external source. The other cluster consists of two state units (recurrent memory for the two-bit action vector), with a fixed weight of 1 from the output layer and a self-recurrent link with a fixed weight μ. These linear units receive input from the output units and from themselves, and act as the mechanism for providing memory to the model. The hidden layer consists of four units. The output layer contains two units, representing

the action vectors. Units in both the hidden and output layer perform a non-linear transform function on their input to produce an output activation in the range 0 to 1. Each layer is fully connected to the next, with recurrent links from the output layer to the state units in the input layer. The network architecture is shown in Figure 5.1 below.

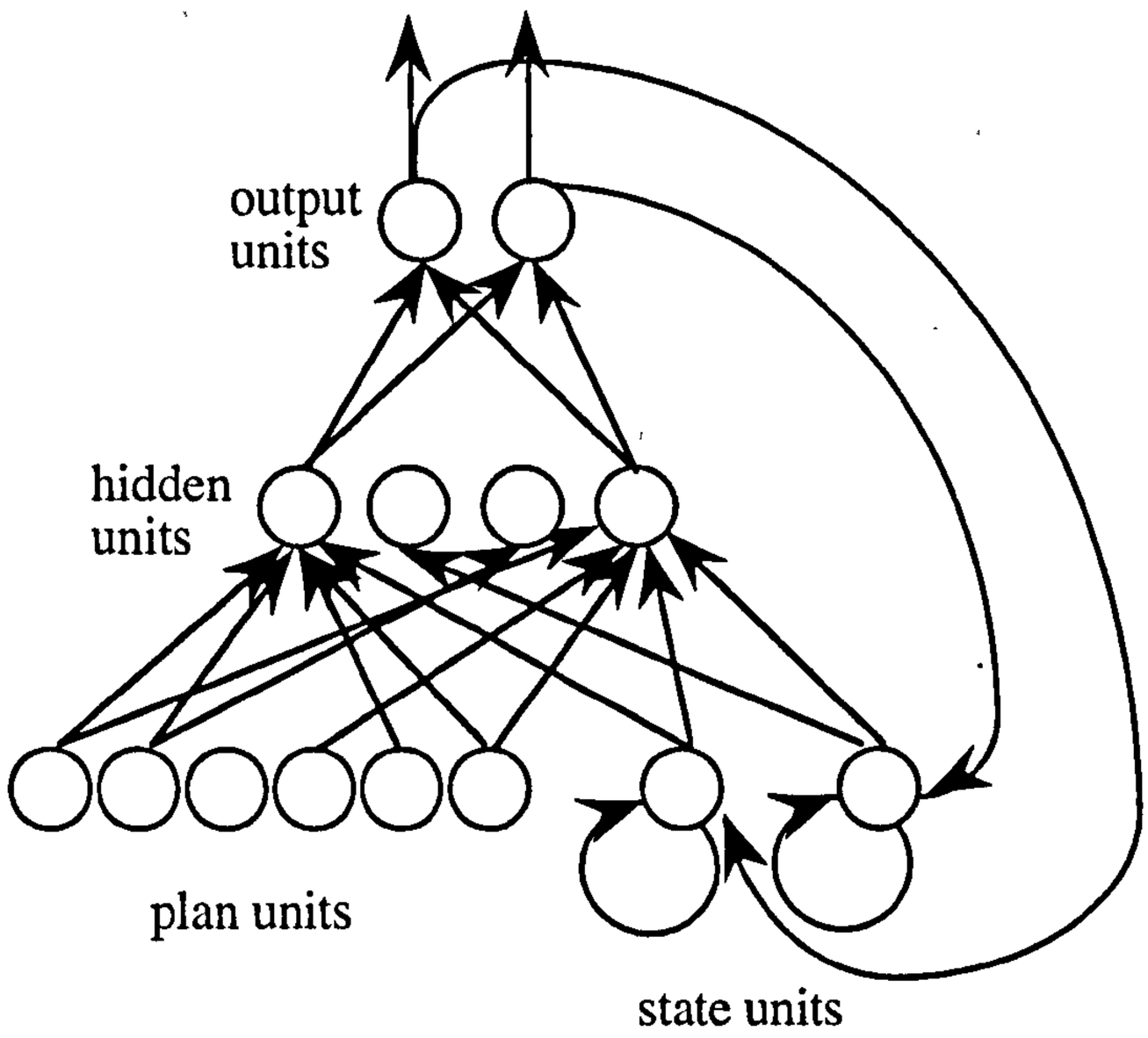

Figure 5.1. The Jordan sequential network architecture for the A[0 1], B[1 1], C[1 0] sequences problem. For clarity some connections have been omitted from the figure, but the general *pattern* of connectivity is as described in the figure.

5.1.1.2 Learning Algorithm

The back-propagation learning algorithm (Rumelhart, Hinton, & Williams, 1986) was used to train the Jordan network. Rumelhart, Hinton, and Williams (1986) found that learning was most rapid with large values of the learning rate and a momentum value of 0.9. The learning rate parameter was set at 0.7 and the momentum term at 0.9, chosen for rapid learning. The μ parameter was set at 0.4, following the value set by Jordan (1986a). The activation function used was the

logistic activation function, with minimum and maximum values of 0 and 1 respectively.

5.1.1.3 Method

Learning each of the six sequences involved the following steps. All the weights in the network were initialised before training by setting them to small random values in the range +0.02 to -0.02. All units in the network were initialised to zero and then one plan was chosen as the input for the plan units. At each time-step in the sequence after the network computed its response to the input, its output was compared to the action in the sequence being learned and errors were propagated backwards according to the above learning algorithm. Following Jordan's method of training, at each time-step the state units were updated with the correct action being learned rather than the computed action, to speed learning. Jordan (1986a) demonstrated that learning still occurs when the state units are updated with the computed action but it takes much longer. During recall, when no learning takes place, the state units were updated with the computed action. The plan units remained active for each time step of the sequence being learned. After each sequence, all units were reinitialised to zero and the above procedure was repeated for the remaining sequences.

5.1.1.4 Results

Five simulations using each representation scheme were run, each of which was started with a different set of initial random weights and was trained on the set of six A B C sequences. Correct performance (as defined by Jordan, 1986a) was achieved when the sum of squared error (SSE) over the output units, summed across all time steps in all sequences, was less than 0.05. For the serial position coding scheme (i.e. where a pair of input units represented each action in the serial position in which it appeared in the corresponding sequence), an average of 356 trials were necessary to

achieve correct performance, and for the transitional coding scheme (i.e. where each input unit represented one of the six possible transitions A-B, A-C, B-A, B-C, C-A, C-B in the sequence being learned), an average of 227 trials were needed. However, more than five simulations of the transitional scheme were run because some of the simulations failed to reach correct performance at all. For these simulations the performance measure oscillated around 1 instead of steadily decreasing towards 0.

5.1.1.5 Discussion

The results above are in accordance with Jordan's results although they differ quantitatively. Jordan found that when the transitional scheme was used, the model reached correct performance (in 98 trials) faster than the serial position scheme (in 129 trials), and this result has been upheld by my own simulations. The results from my own simulations are 227 trials when using the transitional scheme and 356 trials for the serial position scheme. The need for many more simulations using the transitional scheme (even though this scheme achieved correct performance faster) can be explained by the phenomenon of settling in local rather than global minima in connectionist networks (Hopfield, 1982). This essentially means that if a local minimum is encountered during learning then a solution of some sort (usually partial) has been found by the network but it is not the optimal one. The network may become trapped in the local minimum, and if so the optimal solution will never be found and the error will never decrease below that which is found at the local minimum, regardless of the amount of training given. Local minima can be escaped from if there is a degree of uncertainty about the state of the network during training (Hinton & Sejnowski, 1983, 1986). Despite these differences in the results the replicated Jordan Sequential Network appears to function correctly and behaves in the manner expected, and can therefore be used as the basic structure for further simulations.

The Jordan Sequential Network appears to perform well on a number of serial order tasks (Jordan, 1986a) and it seemed appropriate to test it further on a more demanding task of learning serial order. The Morse Code was chosen as a suitable task.

5.2 Learning the Morse Code with a Jordan Sequential Network

The aim of the thesis is to identify the process in speech production of producing a correctly ordered sequence of phonemes given some form of cue (e.g. a lemma) as an identifier. This is undoubtedly a complex process and a suitable task should be chosen to reflect this ultimate goal in a simplified way at least in the preliminary stages of research. The Morse code possesses exactly these properties, i.e. each letter of the alphabet uniquely identifies a variable length sequence (of dots and dashes). This is undoubtedly far simpler than the processes involved in speaking and the simulations to follow start with the Morse code task.

Learning the Morse code involves learning a set of twenty six sequences (one for each letter of the alphabet), all of which are composed of two possible actions, the dot (•) and dash (-) outputs. The sequences vary in length, all being composed of between one and four actions long. Many sequences contain repeated actions, sometimes even sequences of actions. For example, the Morse code for letter D is -•• which contains the repeated action • and the Morse code for letter C is -•-• which contains a repeated sequence of actions, -•. A full listing of the Morse Code alphabet can be found in Appendix I. The task of learning Morse code therefore seemed an appropriately demanding problem for a Jordan Sequential network while possessing the appropriate properties analogous to the topic of the thesis.

5.2.1 Model 5.2

The Morse code problem could not be coded in the same way that Jordan had originally coded his input data for the A B C sequences task. That is, a coding scheme could not be set up where either the input plan for each letter consisted of a pair of units to represent each action in the serial position in which it appeared in the corresponding sequence, or a scheme where each input unit represented a particular transition from one action to another. These schemes were not feasible because the Morse sequences were of variable length and contained many repeated actions and sequences of actions. For this reason, an alternative non-arbitrary plan representation was chosen in which one single input unit represented the plan for each sequence to be learned (i.e. a local representation). The output representation was coded such that there were two (mutually exclusive) units, each representing the '-' and '•' of Morse code respectively. Hence '•' was represented as the vector [1 0] and '-' as the vector [0 1], and for example the sequence for the letter D (-••) was represented by an output of [0 1] followed by an output of [1 0] and finally another output of [1 0].

5.2.1.1 Architecture

The same basic architecture used for Model 5.1 was used again for Model 5.2, with the exception that the input layer consisted of twenty six units to represent the plan for each sequence to be learnt and the output layer which contained two units to represent the actions '-' and '•'. The number of units in the hidden layer was systematically varied in order to give a better chance of finding an optimal model for the problem at hand.

5.2.1.2 Algorithm and Method

Exactly the same procedure was followed as for the original A B C sequences problem used in the simulations of Model 5.1 above.

5.2.1.3 Results

Best performance was achieved by a network with 20 hidden units and these are the results reported here. Simulations using both fewer and more hidden units were run; fewer hidden units (to a limit of about five) generally resulted in a decrease in the number of trials to achieve correct performance but less chance of finding a solution per model simulated, and more hidden units generally gave an inverse result. Correct performance was again defined to be reached when the sum of squared error over the output units, summed across all time steps in all sequences was less than 0.05.

Five simulations of a network with 20 hidden units, each started with a different set of random weights were run. The average number of trials required to attain correct performance was 480. Figure 5.2 shows the rate of change of the sum of squared error during learning for one of the five simulations. Correct performance in this simulation was reached after 222 trials. Beyond this, the sum of squared error continued to decrease towards zero.

5.2.1.4 Discussion

From the results above, Model 5.2 appears to perform well on the Morse code problem. Indeed, all of the simulations of the model found a good solution to the problem. That is to say that none of the simulations found themselves trapped by local minima or unable to find a solution. Figure 5.2 shows an overall decrease in the sum of squared error during training but it is not monotonic. The oscillations of the SSE may reflect the high learning rate used. The learning rate was set very high in connectionist terms at 0.7, a more widely used value is typically 0.1 or less. A smaller learning rate may produce a smoother decrease in SSE during training, but it is also likely to require many more trials.

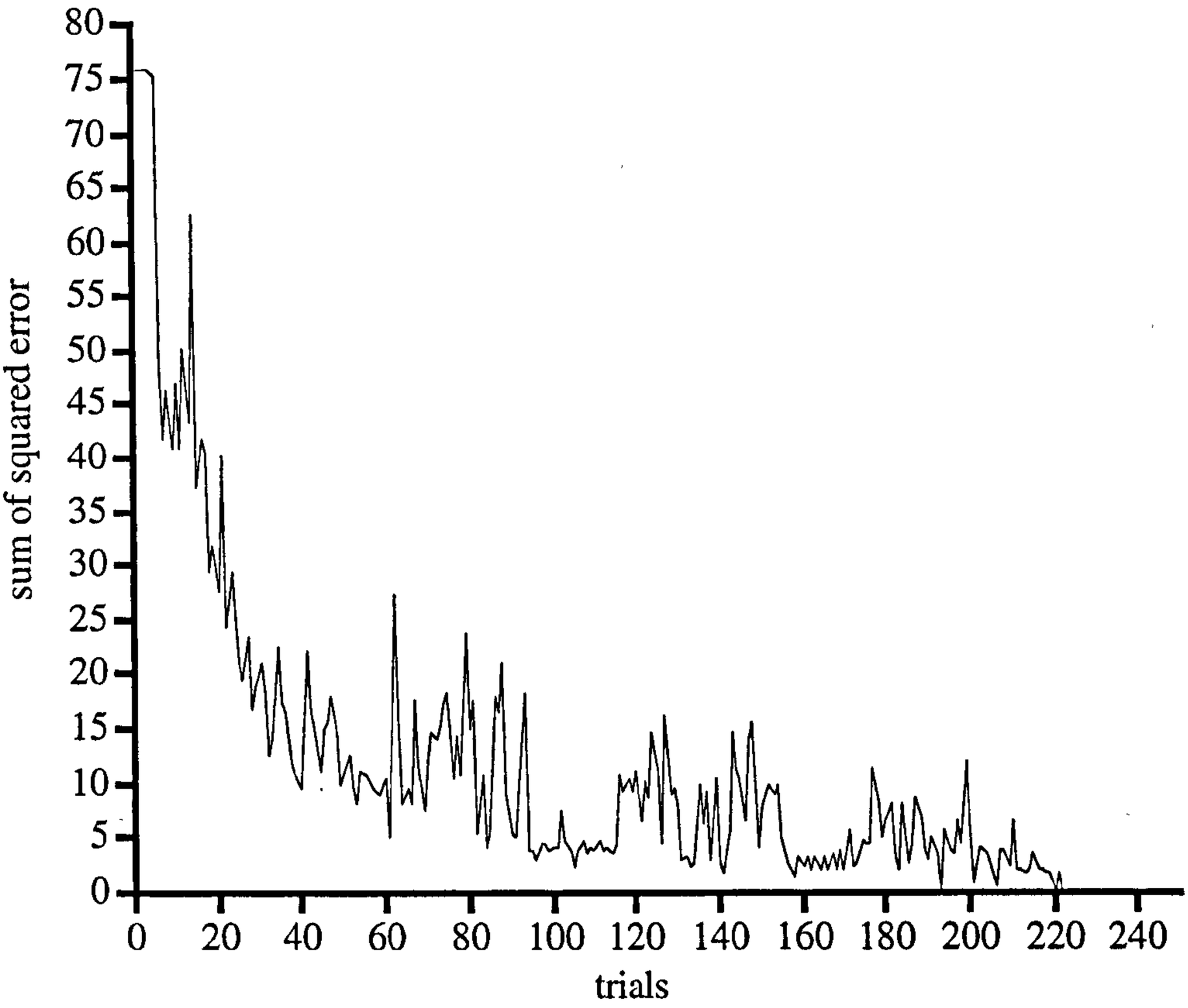

Figure 5.2. The rate of change of the sum of squared error during learning for one of the five simulations. Training beyond 222 trials (point of achieving correct performance) resulted in a monotonic decrease in sum of squared error towards zero.

Although an overall good result was achieved, the performance measure used above may be misleading for the purposes of the Morse code task. Specifically, it does not take into account the output of the model once the end of the target sequence has been reached. This may be of interest given that some sequences (e.g. P •--•) start with a sequence of actions (i.e. •--) that are in itself a complete sequence for a different letter (e.g. W •--). When the actual output produced after the end of the target sequence was inspected it was found to be unpredictable and sometimes introduced ambiguities as to which sequence was really being recalled. For example the sequence for letter L (•-••) was recalled in response to presentation of letter L and E (•); the sequence for letter C (-•-•) was recalled in response to presentation of letter C and K (-•-), and letter D (-••) was recalled for both N (-•) and D. For some

sequences (e.g. X -••-), part of the sequence was (continually) repeated to give a sequence such as -••-••-••-. Other sequences were followed by a null output (both units at resting level) or a continual repetition of a single action (either '•' or '-'). If these observations are considered, performance of the model may not be quite as good as it first appears. It is possible that this problem could be overcome by explicitly training post-end-of-sequence actions as null actions.

It would be desirable to produce a working model that relates in a more analogous manner to a structural design of a model of speech production. The envisaged input to the speech production model is in the form of a set of semantic features to represent words. If the current model architecture were used, then each word (or even each intention) would have to be represented by an individual node. However, the idea of separate representations for each word is not generally supported by the literature, and a representation scheme that reflects (semantic) similarity between words is preferred. This is achieved by using sets of semantic features, or markers (e.g. Hinton and Shallice, 1991) to represent each word.

The current input plan for each sequence is the most simple possible and changing it to reflect the notion of semantic features implies introducing a distributed representation. The resulting coding scheme for the input plan should not impart any information concerning either the actions in the corresponding sequence or their order, as do some of the coding schemes used by Jordan (e.g. as in Model 5.1). Changing the coding scheme to a more realistic form of plan for a model of speech production should not affect performance.

5.2.2 Model 5.3

It seemed sensible to stay with the same task that had already been well performed by Model 5.2 for several reasons. First, because an appropriate comparison between the two models could then be made, and secondly because a

simple task involves fewer variables. It is also easier to understand the workings of the model when completing a simple task, rather than the more complex task of speech production. The input plan for the Morse code problem was therefore adapted for Model 5.3.

This entailed choosing a suitable method of encoding the letters of the alphabet in a distributed form across a number of units rather than designating an individual node to each letter, as in Model 5.2. The visual form of letters can be decomposed into binary pixels in such a way that positions in a pixel grid have different values depending on which letter is currently being displayed. For example, consider the pixel grids in Figure 5.3 which display the letters "S" and "T".

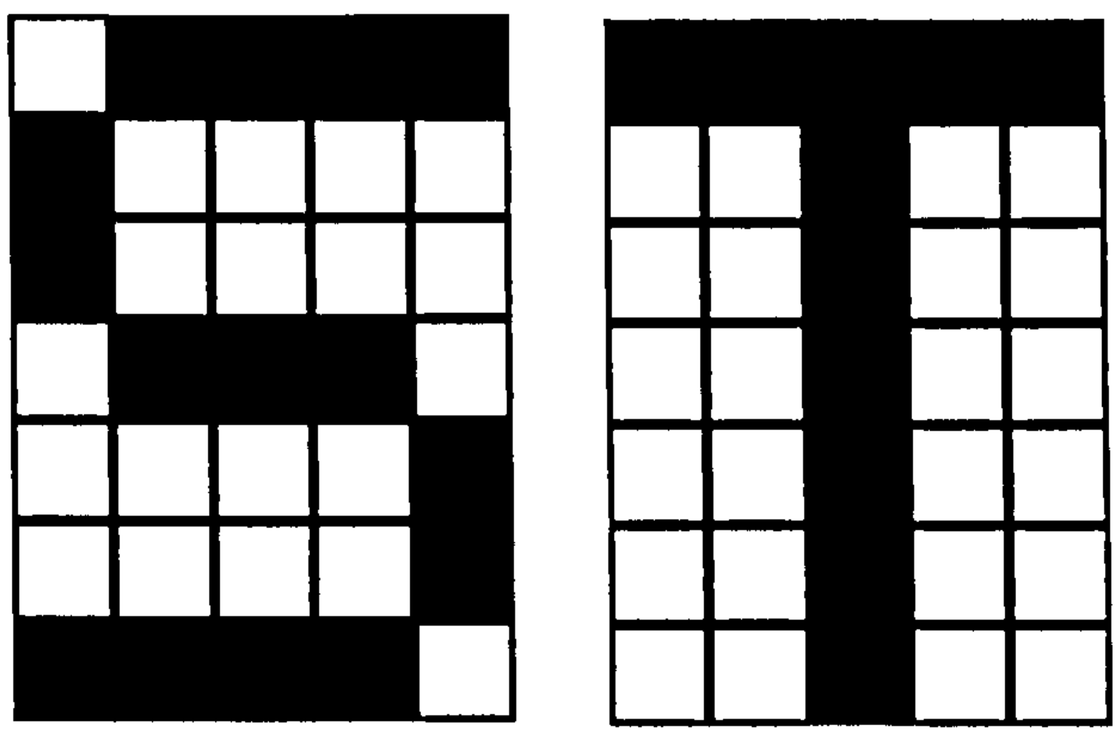

Figure 5.3. An example of a 5 x 7 pixel grid of the letters "S" and "T". A 35-bit binary vector can represent pixel grids by setting each bit to either zero or one, depending on the contents of the pixel grid for each corresponding position in the vector.

The grid is composed of thirty five pixels, arranged so that each letter is five pixels wide and seven pixels high. These pixel grids can easily be converted into binary vectors by setting each bit to either zero or one, depending on the contents of the pixel grid for each corresponding position in the vector. The vector is then a distributed representation of a letter because the letter is coded across many units and

the same vector can represent all the letters of the alphabet. For example in Figure 5.3, the letter "S" would be coded as the binary vector

"0 1 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 0".

The above distributed coding was adopted for Model 5.3 because it was a non-arbitrary scheme already available, but an arbitrary coding would have sufficed because there is no sequencing information contained within each representation. Although the chosen distributed representation of letters stems from a visual domain, this should not detract from its original purpose, that is to demonstrate the robustness of Jordan's Sequential network across an input scheme that is more closely related to a speech production model than the simplest option available.

5.2.2.1 Architecture

The same basic architecture used for Model 5.1 was used again for Model 5.3, with the exception that the input layer consisted of 35 units to represent the plan for each sequence to be learnt and the output layer which contained two units to represent the actions '-' and '•'. The number of units in the hidden layer was systematically varied in order to give a better chance of finding an optimal model for the problem at hand.

5.2.2.2 Learning Algorithm and Method

The same procedure for the algorithm and method as for Model 5.2 were repeated for Model 5.3. It was decided at this stage not to train the model on post-end-of-sequence null actions so that a more direct comparison could be made between the two coding schemes of Model 5.2 and Model 5.3.

5.2.2.3 Results

Five sets of simulations were run of models with different numbers of hidden units. For each set, two simulations with the same number of hidden units were run,

each initialised with a different set of small random weights and the best results obtained from the two simulations for the whole set are reported here. Initially, simulations of a model with 20 hidden units were run, but after disappointing results additional simulations were run of models with 30, 40, 50 and 60 hidden units.

A similar pattern was observed for all the simulations and none of them seemed able to solve the Morse code problem with the distributed coding scheme. For all of the simulations, the sum of squared error oscillated unpredictably throughout training and not one of the Morse code sequences was correctly produced. Figure 5.4 shows the rate of change in sum of squared error for the best in the pair of each of the simulations.

As the graph in Figure 5.4 shows, the sum of squared error in each case oscillated around a very high value (sometimes higher than 88). For example the SSE remained at 82 for the simulations run with 60 hidden units and 62 for the simulations run with 30 hidden units. Even when training was extended to 140000 trials for the most promising simulation (with 30 hidden units) the SSE did not appear to stop oscillating and still remained unacceptably high.

The actual output produced by each of the simulations did not match any of the Morse code sequences. For the simulation with 60 hidden units both output units produced a zero output after 100000 trials which did not represent any legitimate Morse code action at all. The other simulations produced a different output in that for most letters of the alphabet a monotonic sequence was recalled of either '•' or '-', but the two legitimate Morse code actions were never combined in any of the sequences.

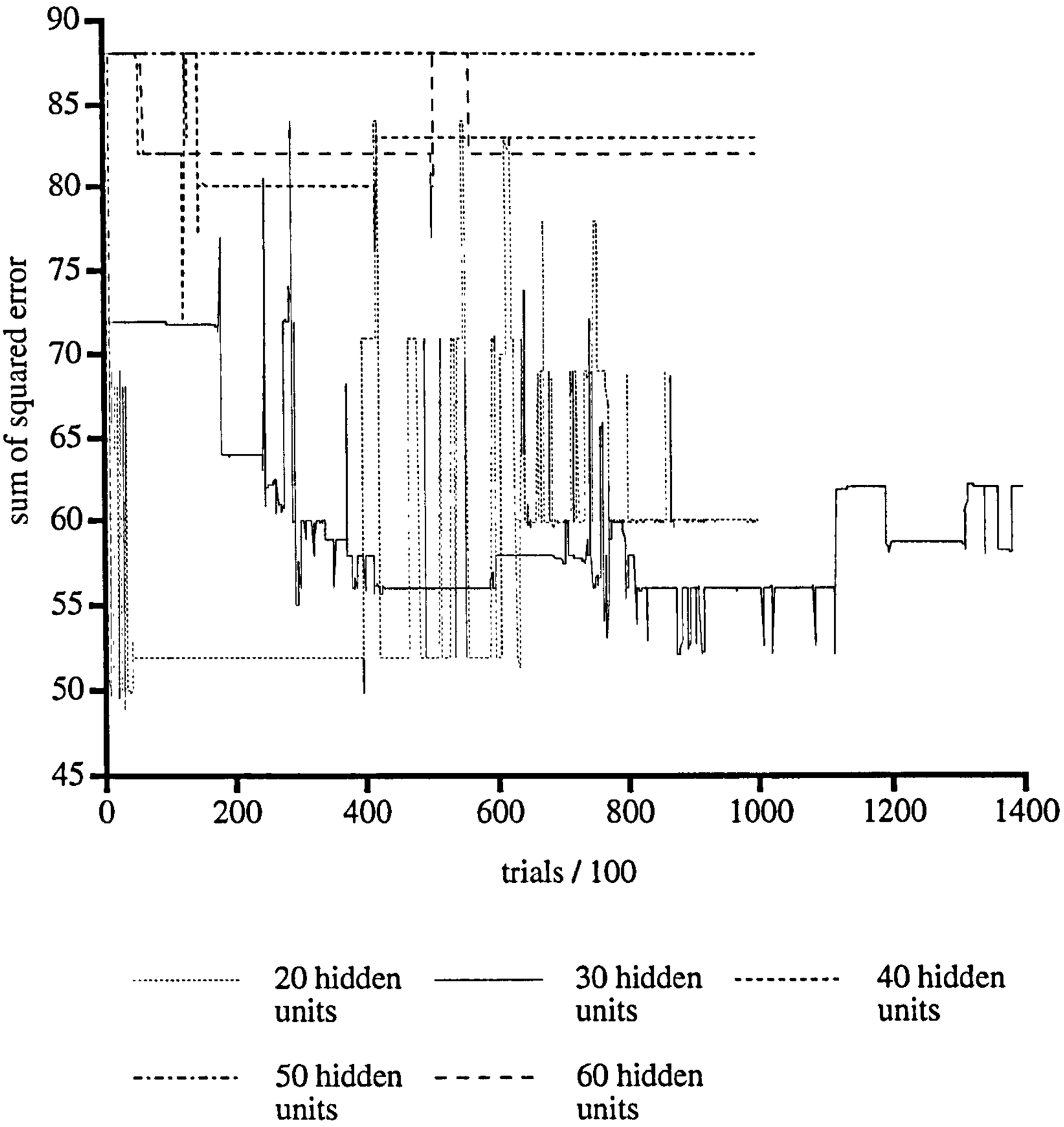

Figure 5.4. A graph showing the rate of change in sum of squared error during training for each of the best of a pair of simulations from a set differing only in their number of hidden units. The sum of squared error was sampled after every hundredth trial of the Morse code problem over 100000 trials. The simulation with 30 hidden units was given extra trials up to 140000 because there was some hope that the error term may actually continue falling.

5.2.2.4 Discussion

The results obtained from Model 5.3 do not support the prediction that changing the coding scheme of the input plan should not adversely affect the results. The distributed representation has clearly had a dramatic affect on performance to the extent that none of the Morse code sequences can be learnt. This is a sharp contrast to

the results from Model 5.2 in which performance was near perfect. This contrast can be seen clearly from Figure 5.5 which plots the SSE curves for Model 5.2 (the local representation) and the best simulation obtained from Model 5.3.

The fact that the only major difference between the two Models was the coding scheme for the input plan suggests that the particular coding scheme used for Model 5.3 is the source of the change in performance. This could be because the patterns on the input units for the coding scheme used in Model 5.3 are essentially arbitrary. That is it is not possible to work out the corresponding sequence of dots and dashes solely from the input plan for each letter.

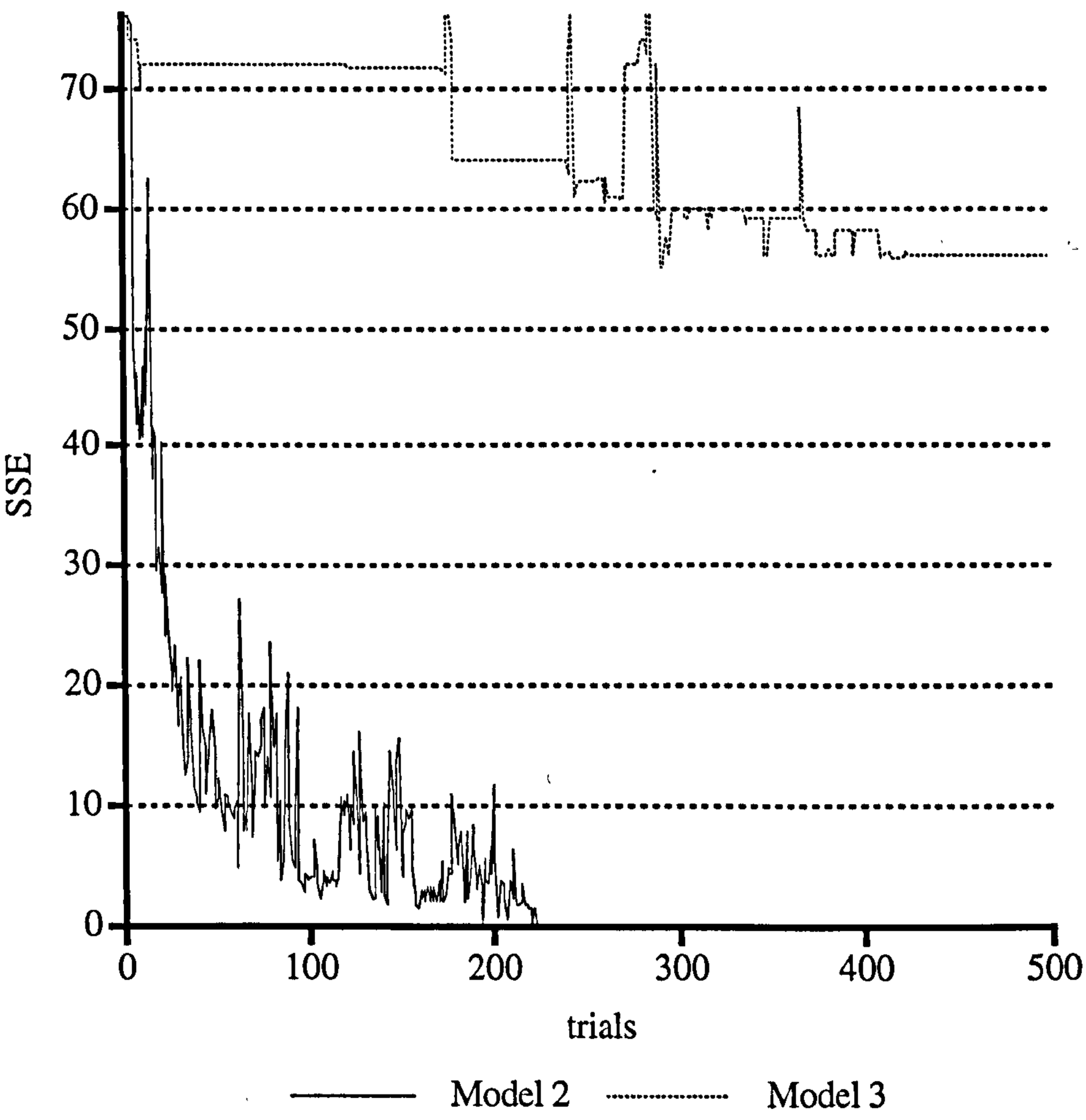

Figure 5.5. A plot of the SSE curves for Model 5.2 (the local representation) and the best simulation obtained from Model 5.3. The number of trials given to the Model 5.3 simulation has been divided by 100 (i.e. sampled every 100 trials) so that the two curves could be displayed on the same axes.

Jordan did use some arbitrary representations of plans for the A B C sequences problem. His results suggest from among the input coding schemes he tried, an arbitrary one was the hardest to learn from. Hence it might be that for some problems a way of encoding the desired sequence in the plan is a necessary requisite for successful learning. The scheme used for Model 5.2 is, on the one hand, essentially arbitrary because the input units do not represent information about the output order of actions. On the other hand, the coding scheme could be argued as non-arbitrary because a single unit represents each sequence and therefore the output can be deduced directly from the form of the input pattern. The need to encode the desired sequence in the plan for the Morse Code problem in a more overt manner does not seem likely.

The scheme used for Model 5.2 treats each plan as independent of all others. The scheme used for Model 5.3 on the other hand allows substantial overlap between different plans. For example, 9 of the 11 plan units used to encode letter I (••) are also used to encode letter T (-, which also used 11 plan units). The similarity of the plans for Model 5.3 may make the sequences much harder for it to learn, hence Model 5.3 may be experiencing a large amount of interference from the input plans during training as it is required to learn different sequences from very similar yet different plans.

However the presence of interference does not necessarily provide a full explanation of the actual output of the Model. If only interference were to blame, then it would not be unreasonable to expect that the most distinct plans should be at least in part learnt. This is not the case because none of the sequences for the Morse Code have been learnt.

An alternative explanation is that because the coding scheme for Model 5.3 involves many more plan units to be switched on a high learning rate will have faster effects on learning than when only one plan unit is switched on as in Model 5.2. Thus

the weights could quickly reach large values, resulting in oscillatory behaviour and an unstable system. This will be referred to as the "too much too soon" hypothesis. If this were the case, choosing a suitably low learning rate would resolve the problems experienced by Model 5.3. This explanation seemed the most promising and was tested in the next set of simulations.

5.2.3 Effects of the Learning Rate

To test whether the learning rate in Model 5.3 was indirectly responsible for the dramatic loss of performance on the Morse Code problem, further simulations of both Model 5.2 and Model 5.3 were run with lower learning rates. It was predicted that if the "too much too soon" hypothesis was valid then the performance of Model 5.2 should be unaffected (although the time to reach correct performance would be greater) and that the performance of Model 5.3 would improve relative to the results obtained with a high learning rate.

5.2.3.1 Learning Algorithm and Method

The same method used in previous simulations was again followed here, with the exception that a learning rate of 0.1 was now used instead of 0.7 as before. The momentum term remained set at 0.9.

5.2.3.2 Results

Ten simulations of Model 5.2 and ten simulations of Model 5.3 were run. In each case, the weights for the simulations were set to small random values in the range +0.02 to -0.02.

Results from the simulations of Model 5.2 were surprising in that not one simulation reached correct performance even after 50000 trials. This is quite different from the simulations of Model 5.2 that used a much higher learning rate of 0.7 where correct performance was always achieved in less than 100 trials.

Model 5.3 showed that most of the time (seven out of ten simulations achieving correct performance) a good solution to the Morse Code problem was indeed found. Of the simulations that reached correct performance, the average time (in number of trials) taken was 4800 trials. Figure 5.6 shows a comparison of the change in SSE of the best simulation obtained of each model.

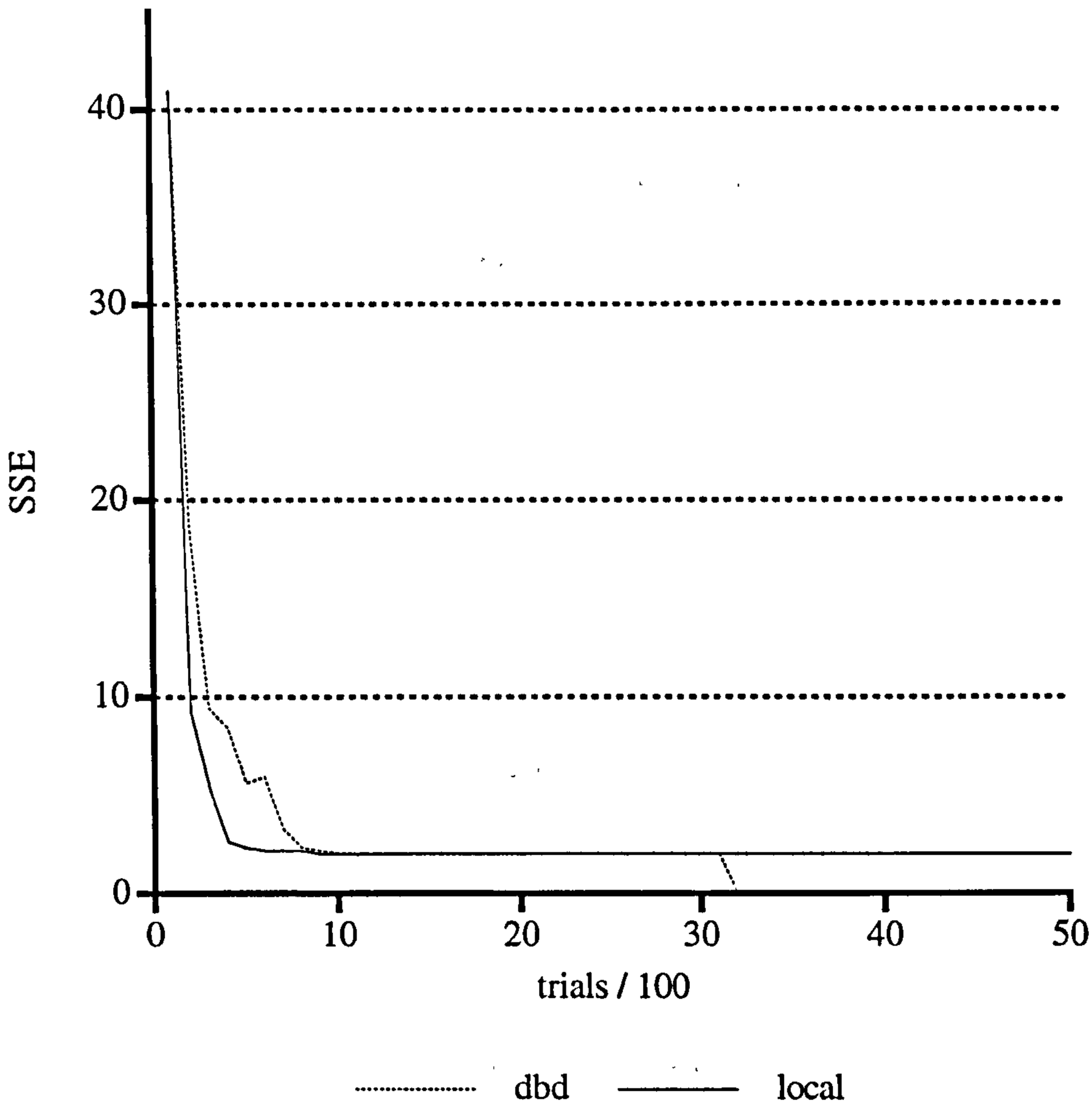

Figure 5.6. A comparison of the change in SSE of the best simulation obtained of Model 5.2 (local) and Model 5.3(distributed). The SSE from Model 5.2 has only been plotted for the first 5000 trials because even after 50000 trials the SSE had only fallen by 0.006.

The SSE curves for the other six simulations of Model 5.3 that found a correct solution are similar to that shown above. The simulations of Model 5.3 that did not find a correct solution did not reduce their SSE below 2.001. The SSE curve for

Model 5.2 shown in Figure 5.6 was a typical result for six of the simulations, the other four simulations of Model 5.2 performed somewhat worse than the one shown in Figure 5.6 and their SSE remained as high as 8.47. Of the six best simulations of Model 5.2, all were able to learn 25 out of 26 sequences correctly.

However, for both Models the actual output produced was sometimes ambiguous as to which sequence was being produced, even though this went undetected by the SSE score. This ambiguity, (also discussed after the results from the first simulations of Model 5.2) arises when sequence initial items are the same for different sequences. For example, the sequence for letter B (-•••) was often also output when presented with letter D (-••). This error is undetected by the SSE score because the sequence for D is identical to the first three items of letter B. The same problems (as for the first simulations of Model 5.2 with a high learning rate) were also detected at post-end-of-sequence output in that subsequent output was unpredictable. Additional simulations were run where Model 5.3 was explicitly trained to terminate each sequence by including a post-end-of-sequence null segment in the training data. For example, instead of learning the sequence (-••) for the letter D, Model 5.3 was trained to produce (-,•,•,NULL). Although Model 5.3 was able to produce each letter without ambiguity, there was a still a problem with post post-end-of-sequence output because the Model had learnt to produce the null segment just as part of each sequence rather than as sequence termination. The model still produced a cyclic repetition of the sequence, i.e. -,•,•,NULL,-,•,•,NULL, etc.

5.2.3.3 Discussion

The results provide support for the "too much too soon" hypothesis. The coding scheme for Model 5.3 involved many more plan units to be switched on and so the result of a high learning rate took effect sooner and to a greater extent than when only

one plan unit was switched on as in Model 5.2. This was further supported by the results from Model 5.2 that showed a high learning rate can still be used when only one plan unit is switched on at a time. Thus the more plan units that are switched on at any one time, the lower the learning rate needs to be to find a correct solution, and decreasing the number of plan units that are switched on increases the optimal value for the learning rate.

5.3 Conclusion

The aim of this chapter was to explore the viability of the Jordan Sequential network (Jordan, 1986a) with respect to its ability to provide a basic mechanism of serial order that could be used in a model of speech production.

Model 5.1 demonstrated a simple implementation of Jordan's Sequential network capable of replicating results reported in his original study. Quantitative differences were found between the results reported from Model 5.1 above and Jordan's original work, although this was to be expected. An unexpected result was the presence of local minima, which was unreported by Jordan.

Model 5.2 was designed to test Jordan's model further by providing a harder task for it to perform, but one which also had some relation to the sequential output of phonemes in speech production. Hence the Morse code problem was introduced because it shared some similarities with speech production but was also a more tractable problem. A simple representation of the plan for each letter of the alphabet (a local representation) was given and initial results appeared excellent. However further analysis of post end-of-sequence output revealed ambiguities and inconsistencies in the sequences produced by the model. Nevertheless, the model had performed well on what was quite a hard task to learn.

Model 5.3 differed from Model 5.2 in that a more realistic representation of plans for each letter of the alphabet was used. The scheme was chosen such that it should still not provide any information about the associated sequence directly in its coding (i.e. it was an arbitrary coding scheme). Results were disappointing and a solution could not be reached by the Model. The performance measure often oscillated wildly in the search for a global solution that it never found. Further simulations of both Models 5.2 and 5.3 revealed a trade off between the number of active units in the input layer and the optimal value of the learning rate such that as the number of active input units increased, the optimal value for the learning rate decreased. A suitably low learning rate for Model 5.3 yielded a correct solution 70% of the time.

Jordan's (1986a) sequential network appears to perform at its best far less than satisfactorily (at 70% correct) on the Morse Code problem providing that a suitable set of model parameters (e.g. the learning rate, momentum, size of initial weights etc.) can be found. However ambiguities arise in the post-end-of-sequence output produced by the model which do not appear to disappear when it is explicitly trained to terminate each sequence. This sort of behaviour is not akin to normal speech production because speakers do not usually produce extraneous phonemes after having reached the end of a spoken word. The reliability of the model may also be questioned on the grounds of its success rate only reaching 70% (i.e. 7 out of 10 models could find a solution) on what is a very small set (26) of sequences.

The question of the reliability of recurrent neural networks has been raised before (Chater, 1989; Chater & Conkey, 1994). Chater (1989) found that the copy-back training method was far inferior to the back-propagation-through-time algorithm. He found, contrary to the theoretical analysis of Beaufays and Wan (1994), that the copy-back algorithm often failed to learn tasks that could be learnt using back-propagation through time. Chater (1989) also noted that both methods

would suffer as the dependency on information in the sequence became more distant. Although, according to Chater (1989), back-propagation through time survives somewhat better than copy-back as the task complexity increases, it also becomes less feasible to implement because of the computational expense of the method. It also has the undesirable property that past states of activation must be remembered, which is tantamount to the same requirement of temporal window models for some sort of buffering of past states. The simulations in this chapter do not address the debate over the equivalence of back-propagation through time versus copy-back. They are however in some ways consistent with Chater (1989) in that the copy-back regime is often unreliable. In light of the limitations of this approach, it did not seem wise to implement Jordan's (Jordan, 1989; Jordan & Rumelhart, 1992) more advanced sequential model which was based on his original model, nor to apply the model to a speech task involving long sequences, and attention was instead given to an alternative model, described in detail in the next chapter.

Chapter 6

6. Recurrent Adaptively Parameterised Error Correcting Systems (REAPECS)

The recurrent network topology developed in this chapter was inspired by McLaren's (1993) feed-forward connectionist model. Some of the reasons for using distributed representations in a connectionist model of speech production have been given in the previous chapters. One reason is that the similarity of speech sounds can be better described if each phoneme is represented by a set of phonetic features. However, one of the limitations of models that use distributed representations with the error based gradient descent method of learning is that the model can only retain all it learns if all the items to be learnt are interleaved in a single training set which is then repeatedly presented to the model (McCloskey & Cohen, 1989). If the model is instead trained to perfection on each training pattern before the next is encountered then a quite different result is obtained; learning new items often causes previously learnt items to be forgotten. This effect is known as *catastrophic interference* (McCloskey & Cohen, 1989). This is not characteristic of the acquisition of many human cognitive skills. For example children learn to speak by first learning basic speech sounds, then simple words and so on.

This limitation of one of the most desirable qualities of connectionist models lead McLaren (1993) to develop a model that could learn information in a sequential manner without losing previously learnt associations, in a way more akin to the acquisition of information often displayed by humans. Thus McLaren's (1993) model

displayed certain qualities that could provide a good basis for a model of serial order if it could be extended to learn sequences.

In this chapter a brief description of the main principles behind McLaren's model is followed by an explanation of how it could be extended to a model of serial order. These ideas are then put into practice with a series of simulations that develop a recurrent version of McLaren's model. The model is tested on the same Morse Code tasks set for the Jordan sequential network in the previous chapter.

6.1 McLaren's (1993) Model

The heart of McLaren's (1993) Adaptively Parameterised Error Correcting System (APECS) is its capacity to prevent established associations from being modified (and risking becoming unlearnt) by associations learnt at a later time, by the selective control of the bias and learning rate parameters which it assigns to each hidden unit. The algorithm is based on the error based gradient descent method, but differs in that hidden units are selected to mediate certain mappings and once a hidden unit develops weights to mediate a mapping from input to output, those weights will not be altered by training on subsequent mappings regardless of whether they partly overlap. For example, suppose in the diagram in Figure 6.1, two mappings are to be learnt, first mapping A and then mapping B.

First, mapping A is presented and a hidden unit is selected to mediate the mapping. This is indicated in Figure 6.1 by the unit labelled A in the hidden layer. The unit develops strong positive links to the appropriate output and input units until the mapping is sufficiently well learnt, indicated by the bold lines in Figure 6.1. Mapping B is presented next which shares some of the units that represent mapping A. Although the same hidden unit that is now part of mapping A will become partly activated it will not be selected to mediate mapping B and hence the weights mediating mapping A are not changed. Instead, the bias of the hidden unit mediating

mapping A is adjusted so as to inactivate it when mapping B is presented, but not mapping A. A different hidden unit is selected when mapping B is presented (labelled B) which again develops strong positive links to the appropriate input and output units. Although in the example only one hidden unit mediates each mapping, this is not strictly the case in McLaren's model as more than one hidden unit can be used to mediate a particular mapping.

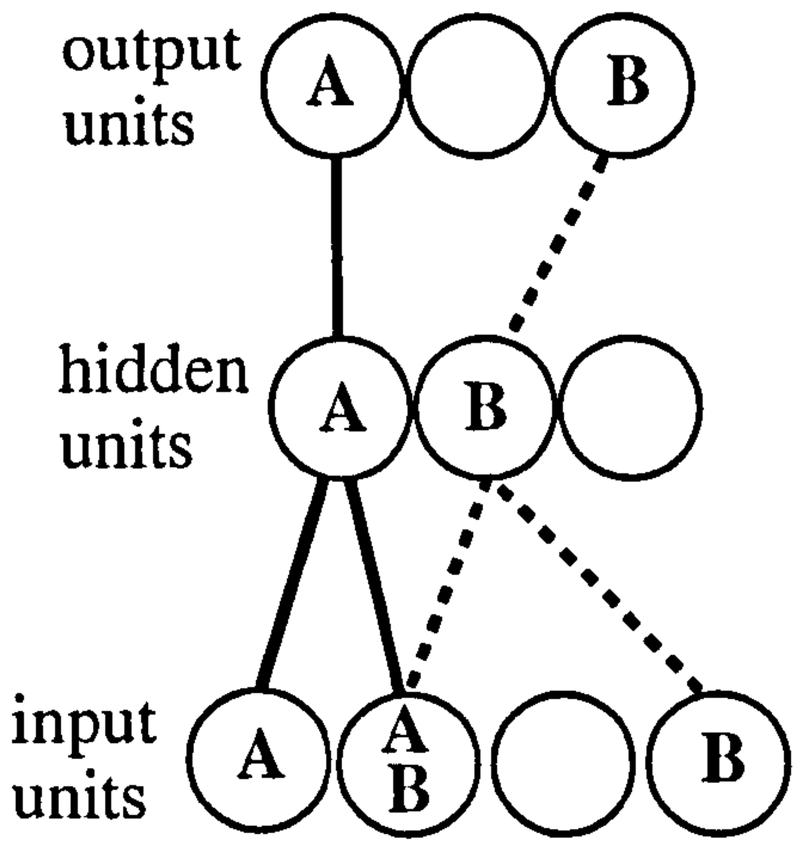

Figure 6.1. A simple APECS model. First mapping A is learnt, depicted by units containing the label A and connected with bold lines, and then mapping B is learnt (units with label B and connected with dashed lines) without disrupting the weights mediating mapping A by selecting a different hidden unit to mediate the mapping.

One of the assumptions made by APECS is that each mapping must be uniquely identifiable for it to be successfully learnt. According to McLaren (1993), if each input-output mapping could be uniquely identified, then any training set could be learnt and retained without suffering from interference from subsequent training on different (but maybe similar) mappings. If APECS could be applied to learning sequences such that each item within a sequence could be uniquely identified and hence protected from interference, and if each state of the sequence could be somehow connected together in the appropriate order, then any sequence, including complex ones containing repeated sub-sequences, could be successfully learnt using

the above principles. Specifically, if each sequence could be identified in some way by a unique initial cue to disambiguate each sequence, then provision of an appropriate recurrent network architecture and methodology would ensure that representation of elements on subsequent time-steps would also be unique yet connected as a sequence and hence learnable by APECS. This would be possible because a specific hidden unit (or units) would be selected to carry the (by definition) unique initial mapping in the sequence. A recurrent topology would enable the chosen hidden unit(s) to form the input at the next time-step and because that hidden unit(s) is unique to the currently learnt mapping, this guarantees that the input on the next time-step will be unique as well. This guarantees that the next hidden unit selected will be specific to this mapping, and so on. As the sequence is represented by a set of hidden units unique to that sequence, indeed to each particular mapping in that sequence, interference between sequences would be minimal and repetitions of any length could then be accommodated. Hence a recurrent version of APECS (REAPECS) could learn sequences.

6.2 Recurrent APECS (REAPECS)

The basic idea of REAPECS is the same as that of APECS: to govern learning by controlling which hidden units become active for each mapping in the data set so that a different hidden unit(s) is active for each one. This is why each mapping must be unique. Then the strengths of the connections to and from each hidden unit can be selectively determined and inappropriate activity on the hidden layer can also be controlled by setting the bias levels accordingly. In order to accomplish this, each hidden unit has its own learning rate parameter which effectively controls only the weights to which it is connected. To prevent hidden units from becoming inappropriately active when a similar input pattern is presented, the bias is adjusted rather than the weights feeding into that unit.

Figure 6.2 provides an outline of how the principles of recurrent networks could be extended to APECS to learn Morse Code sequences.

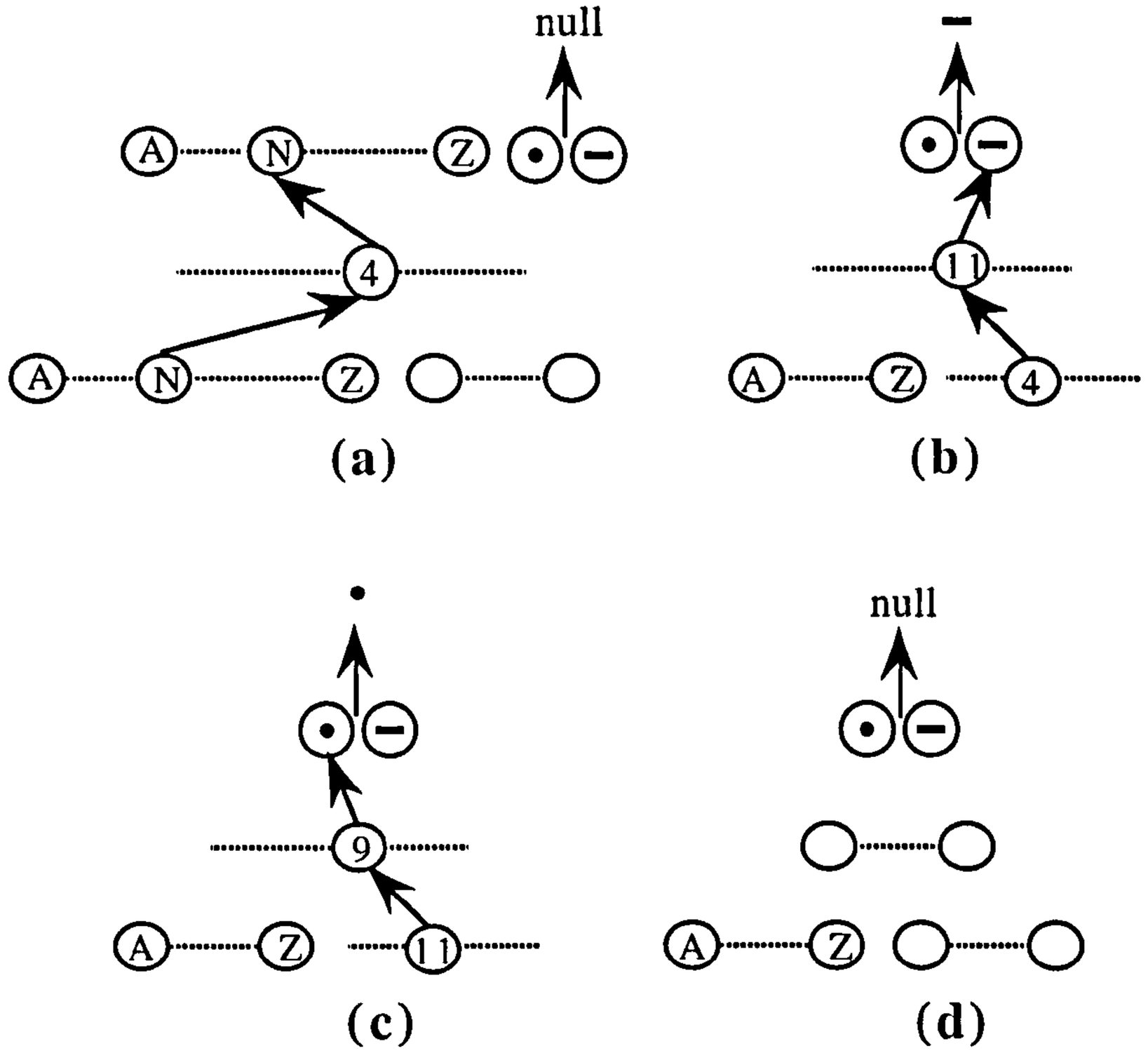

Figure *6.2.* Learning the Morse Code sequence for the letter N. Annotated circles represent units in the network, horizontal dashed lines indicate additional units not shown. The top layer represents the output units, the middle the hidden units and the bottom the input units which are separated into two clusters. Lines connecting units between layers indicate strong positive links developed during training and numbers within each unit serve to distinguish different units. The arrows denote the direction of flow of activation in the network. At each stage in learning the sequence for letter N, a different hidden unit representation develops which is then transferred to the copy-units on the next time-step.

For the initial time-step, (a), the model learns a unique cue for the sequence to follow. This is mediated by an available hidden unit and the links develop accordingly. On the next time-step, (b), a different hidden unit(s) is chosen to mediate the first element (-) in the sequence. The input pattern for letter N is no longer active, but copy-unit (number 4) has a high activation because it was chosen on the previous time-step to mediate the initial cue for letter N. Strong links are

developed from this unit and to the '-' unit in the output layer. The logic of time-step 2, in (c), is the same as for (b). On the last time-step (d), no links develop because the end of the sequence has been reached and there is no active unit in the target output and therefore no hidden unit(s) becomes selected.

The models in the following simulations implement a REAPECS model, whose task description remains the same as for the Jordan Sequential network, that is learning the Morse code.

6.2.1 Model 6.1

Model 6.1 was designed to test whether or not APECS could be extended to a task that involved recalling sequences. The Morse code problem, for the purposes of these computer simulations is defined as the task of responding with the appropriate sequence of dot's (•) and dashes (-) when given the corresponding letter of the alphabet as input. Letters of the alphabet were represented in a simple manner by using a single unit for each letter. Note that in this case because letters are represented by a single unit there is no potential interference between input patterns. Hence Model 6.1 was implemented as a simple demonstration of the extension of the APECS model to a recurrent topology so that sequences could be learnt.

6.2.1.1 Architecture

The model architecture is a variation of the recurrent network topology, consisting of groups of units arranged into input, hidden and output layers. The input layer consists of two clusters of units. Twenty six units in one cluster encode each letter of the alphabet so that one unit is solely dedicated to representing each letter. These units are set by an external source. Copy units form the other cluster. Their activations are derived from the activation of the units in the hidden layer on the previous time-step. These units act as the mechanism for providing memory of

previous events to the model in a similar way to Jordan's sequential network. They provide within-sequence context and are also referred to as context units. The hidden layer contains one hundred and fifty units. There must be enough hidden units to carry all possible mappings in the data set because once a hidden unit is selected to carry a mapping it cannot be used for another. This number was therefore based upon the number of input-output mappings necessary to encode the full alphabet of the Morse code. The output layer contains three clusters of units; twenty six units to identify each letter of the alphabet (as in the input layer), two units to represent the dot (•) and dash (-) output of Morse code and another cluster of copy units whose *target* activations are also derived from the activation levels in the hidden layer on the previous time-step. They serve a similar purpose to the copy units in the input layer for providing context and again are referred to as context, or copy units. Figure 6.3 illustrates the architecture of the model.

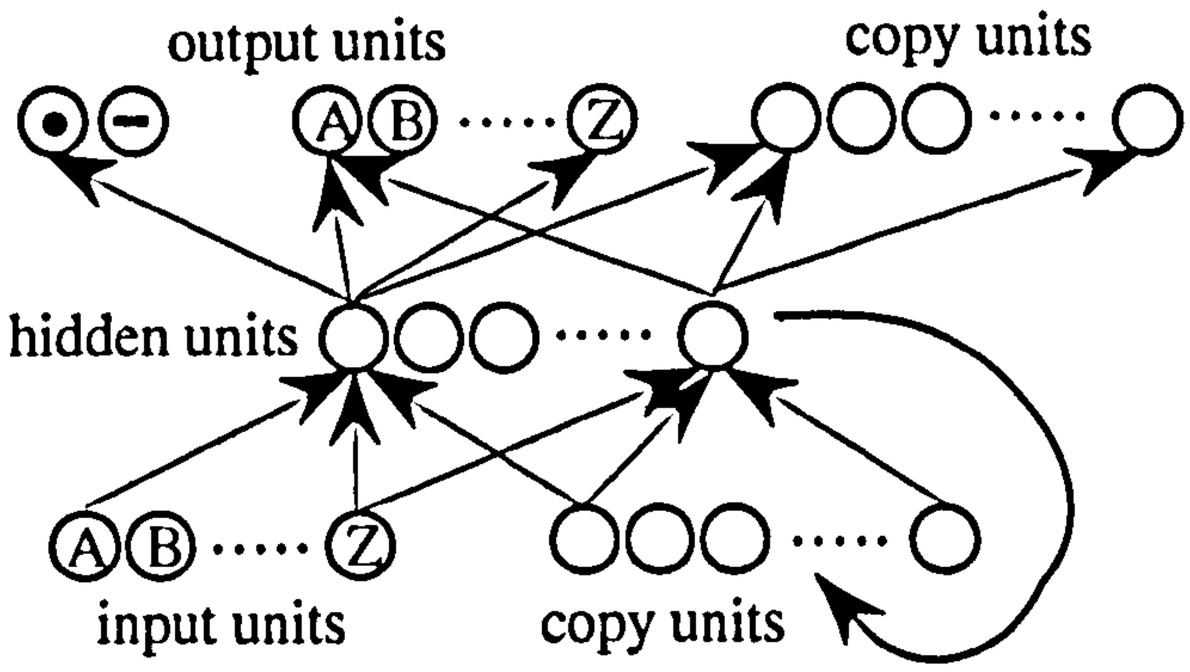

Figure 6.3. The network architecture for the REAPECS Morse Code model.

Both sets of copy units are required by the recurrent network topology running REAPECS to produce sequential output. The copy units in the input layer are similar (functionally speaking) to the state units in Jordan's (1986a) sequential network; they provide the mechanism that allows items to be recalled in a consecutive manner to form a sequence. Their activations are derived from the hidden units' activations on

the previous time-step and they essentially form the link between each item in the sequence being learnt. The copy units in the output layer help to identify uniquely each mapping in the sequence. Without these copy units, unique identification would be problematic at best and sequences with repeated elements could not be learnt. The input units and both sets of copy units perform a linear transform function. All other units perform a sigmoid transfer function on their input, producing an output activation in the range 0 to 1.

6.2.1.2 Learning Algorithm

Considering that the copy-back regime is less computationally expensive and imposes no a priori knowledge on the length of the sequence to be learned, it was favoured above the back-propagation through time regime for the purposes of these simulations. The learning algorithm (McLaren, 1993), Adaptively Parameterised Error Correcting Systems (APECS) has been adapted in the style of the back-propagation copy-back training regime. It differs from the standard back-propagation algorithm by competitively selecting one hidden unit to carry the mapping for each input-output pair. The hidden unit selection procedure originally developed by McLaren has been replaced with an optimised version which is less computationally intensive, but achieves the same effect. As a full exposition of the original version is available elsewhere (McLaren, 1993), only the optimised version will be given here.

Learning occurs in two passes, in the same manner as in back-propagation. The forward pass occurs first in which the network generates an output activation pattern in response to the cue for the sequence to be learnt. Learning the target response occurs during the backward pass which commences immediately after the forward pass. Up to now, the procedure followed matches that of standard back-propagation.

The characteristic feature of REAPECS that sets it aside from standard error back-propagation is the dynamic control of individual learning rates for each hidden

unit and the appropriate adjustment to the hidden unit biases. In this case, the learning rate for a given hidden unit is determined by its raw error score. The raw error score of a hidden unit is the sum of all the error terms of the output units to which it is connected. REAPECS selects and controls hidden unit activation in the following way:

(i) All hidden units are assessed for their suitability to mediate the particular part of the sequence being learnt without interfering with previous mappings. Each hidden unit is given a suitability score, which is equal to its raw error score. Hidden units whose raw error score contains a significant negative error component (the significance level is determined by the *veto* parameter) have their suitability score set to zero. This is because a large negative raw error score signifies that the hidden unit already mediates a mapping. A hidden unit is otherwise potentially suitable. This can be explained in detail by referring back to Figure 6.1. In the Figure, first mapping A is learnt and the hidden unit labelled "A" carries the mapping. When mapping B is presented, output unit "A" will automatically become activated because the input patterns for mappings A and B overlap. The error score of output unit "A", calcuated according to (6.1), where err_i is the error score of output unit i, t_i is its target activation, and a_i is its actual activation, will be negative since the activation of output unit "A" is greater than its target activation.

$$err_i = (t_i - a_i)a_i(1 - a_i) \tag{6.1}$$

Next, the raw error score of each hidden unit is calculated, which is simply the sum of the product of the error score of each output unit and the weight

connecting it to the hidden unit, for all output units. Going back to Figure 6.1, hidden unit "A" will receive a large negative component from output unit "A" because the error score of output unit "A" is negative and there is a strong positive weight connecting it to hidden unit "A". This indicates that hidden unit "A" is not suitable because it already mediates a mapping. However, the other hidden unit in the Figure will not receive a large enough negative component from output unit "A" because there is no large weight between itself and output unit "A". Hence it can be chosen to mediate mapping B.

The parameter called veto controls how sensitive REAPECS is to determining the suitability of hidden units to enter the competition to mediate mappings. The more negative the veto parameter, the less sensitive the suitability criteria becomes (i.e. more units are likely to be deemed suitable to compete to carry the mapping), and the smaller the value the more sensitive it becomes (i.e.. fewer units are likely to be accepted as suitable). The veto parameter was set to -0.05.

(ii) The selection score for each hidden unit is then calculated as a function of its suitability score and its current activation. A hidden unit is then selected to mediate the particular mapping at hand on the basis of its selection score. The unit with the highest score is selected.

(iii) All other hidden units remain un-selected.

(iv) The learning rate of the selected hidden unit becomes non-zero. The value of the learning rate was set to 1. This applies to the learning rate controlling the changes in the weights into and out of that hidden unit, but not to its bias.

(v) The learning rates of the non-selected hidden units remain set at zero.

As a general guideline, for most connectionist models, very small learning rates make the process of learning slower but large learning rates often result in incorrect solutions. However, one of the advantages of the APECS algorithm over standard back-propagation is that hidden unit representation does not overlap between mappings. Therefore larger learning rates can be accommodated because learning the weights for any particular mapping will not interfere with weights used for other mappings.

Once an appropriate hidden unit has been chosen to learn the mapping, the weights are modified in the normal way as for back-propagation. (The weight updates for the connections between the input and hidden layer are moderated according to the number of active input units to keep the size of the weights under control. This is achieved by dividing each weight update by the total number of active input units.) After the weights have been updated, the bias of all hidden units is updated. Note that if a hidden unit was not selected, only its bias can change. The bias adjustments are determined by the following factors:

(i) If the raw error score (err_i) of the hidden unit is negative, then its bias increment is calculated according to (6.2).

$$\Delta \text{bias}_i = \text{err}_i \times T \tag{6.2}$$

where T is a constant, which will result in an actual decrease in value to its bias because err_i is negative. Hence a large negative bias will develop.

(ii) If its raw error score is positive its bias increment is calculated according to (6.3),

$$\Delta bias_i = err_i \times (1 - \alpha_i) \times K \tag{6.3}$$

where K is a constant. This leads to a small increase in value to its bias unless the units has a high learning rate (α) (i.e. it has been selected to mediate the mapping) in which case its bias is unaltered.

(iii) All biases are subject to the constraint that they cannot take a positive value and are set to zero if the update rule results in assigning them to a positive value.

The effect of the above procedure is that if a hidden unit was inappropriately active for a particular mapping then a large negative bias will develop to prevent it from becoming active in future when that input mapping is presented, while the bias of the selected hidden unit remains unchanged. Both passes are now complete.

6.2.1.3 Method

The model was initialised by setting all the network weights to small random values within the range -0.02 to +0.02. Training the model involved presenting each letter of the alphabet as input by setting the activations of the input units appropriately and then performing the forward pass and backward pass of the learning algorithm. The appropriate Morse code sequence was provided as the desired output. Each element of a Morse code sequence to be learnt was presented on consecutive time steps. The pattern of activation on the input layer together with the pattern of activation on the output layer defined a mapping. Training on the data set consisted of cycling through each letter-to-Morse-code sequence pairing, learning one time-step of the sequence (i.e. a mapping) at a time. A consequence of the APECS learning algorithm is that each mapping must be sufficiently well learnt in one presentation in

order that the hidden unit previously selected to mediate it be protected from being selected for similar mappings later on in training. This was achieved by applying consecutive iterations of the learning algorithm to each mapping 1000 times. For the first time-step during the training regime the model was given a letter as input and trained to produce the same pattern for that letter as the desired output. This uniquely identified a cue for each sequence by ensuring a unique hidden layer representation was created for each letter. For this initial step in the sequence, the copy units in the input layer were set to zero. On subsequent time-steps, a pattern of activation was copied from the hidden units to the copy units in the input layer and the model was trained to produce the appropriate part of the Morse code sequence for that letter as the desired output. The input units were set to zero after the initial step in learning and remained so for the duration of that sequence. During the learning phase, the hidden units whose activation levels were 0.6 or higher were copied to the copy units in the input layer at the end of each time step. However, during recall, the activation levels of all the hidden units were copied without modification to the copy-units in the input layer. For the last time-step in the sequence the model was trained explicitly to produce a null response. Thus when the model generated sequences during recall, the end of a sequence could be detected by the absence of any output. For all time-steps, the target output of the copy units in the output layer was derived from the activation levels of the copy units in the input layer. The input sequence of letters was completely random for each trial.

The performance of the model was measured by first computing the sum of squared error (sum of the difference between the desired and actual value of the output units squared) for each time step of each sequence and summing over the whole training set. The final measure was obtained by computing the mean squared error (MSE), which was the average value for all mappings in the training set. This

value was calculated after each trial, where a trial is defined as one complete pass through the entire training set. The MSE is mathematically expressed in (6.4),

$$\mathrm{MSE} = \frac{\sum_{j=M}^{j=1}\left(\sum_{i=n}^{i=1}(t_i - a_i)^2\right)_j}{M} \tag{6.4}$$

where M is the number of different mappings in the training set, n is the number of output units, t_i is the target activation of the i^{th} output unit, and a_i is the actual activation of the i^{th} output unit. The numerator of the above equation corresponds to the sum of squared error as defined by Jordan (1986a). However, it should be noted that the numerator term is not directly comparable with the equivalent Jordan network in chapter 5 because the number of units in the output layer is different here. Performance was also assessed in terms of how many correct sequences the model was able to produce out of a possible total of 26. The output from the dot and dash units were read as binary values; a unit whose activation was above 0.6 was read as 'on' and 'off' otherwise. A response was thus defined as correct when the appropriate output node was 'on' and the other 'off' (thus defining either a '•' or '-' response) for a particular item in a sequence. Note that if a unit receives no net input then it will still have an output activation value of 0.5.

6.2.1.4 Results

Ten simulations of Model 6.1, each with different starting weights were trained on the same data set. The MSE values during training were very similar across the ten simulations, and the results presented are the averaged MSE scores from all simulations.

The average MSE of all the simulations before they were trained was 0.1151813. This fell over four trials to 0.00027628, after which time all simulations had learnt the entire set of sequences for the alphabet. Figure 6.4 shows the rate at

which the average MSE of all simulations fell after each trial, compared with the rate at which the average number of sequences correctly recalled increased.

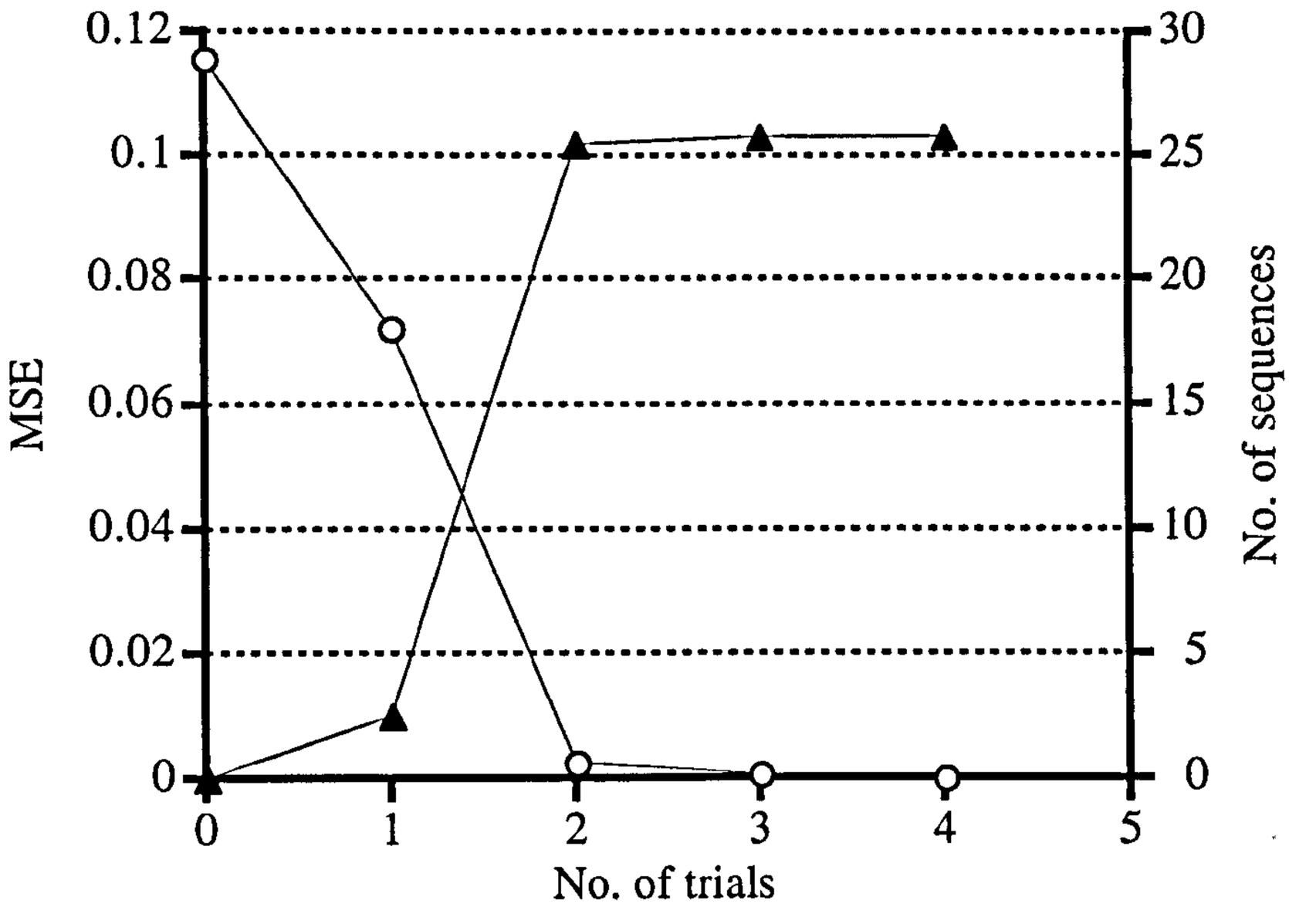

Figure *6.4.* A graph of the average performance of ten simulations of Model 6.1, each assessed and then averaged after the completion of each trial (of a total 4). The left ordinate plots the averaged MSE of the ten simulations, and the right ordinate plots the average number of correctly recalled sequences, against the number of successive trials as abscissa.

The average MSE before training is already very low. This is not surprising given that the default value of an output unit in a REAPECS model is to be switched off unless it appears as part of a mapping in training (in which case its incoming weights are increased to switch it on). Before any training occurs, all output units will assume their default values when presented with the data set. There are a total of 28 output units, only one of which should be active for any particular mapping in the data set. Therefore when calculating the MSE before training there will only be one unit per mapping that will provide a significant contribution to the MSE score.

The graph in Figure 6.4 illustrates how quickly good performance can be achieved using REAPECS. After one trial of the training data, there is a reasonable decrease in MSE and already the model has begun to correctly recall a few sequences. However, the most impressive increase in performance comes after the second trial, when on average the model is able to recall all the sequences and the MSE has reduced to 0.00246. Hence after just two trials through the training data the model's performance is already sufficient to correctly recall the Morse Code sequence for any letter of the alphabet.

6.2.1.5 Discussion

Such good results clearly demonstrate the model's ability to learn sequential information. The defining feature of REAPECS that sets it aside from other models is the way in which it controls activity in the hidden layer of units. Once a hidden unit has been selected to carry a particular (and unique) mapping during training, strong positive links are formed to the appropriate input and output units so that presentation of the input pattern will activate the correct output pattern. The same hidden unit is never re-selected to mediate a different (or even similar) mapping. The network effectively forms unique representations at the hidden layer, capturing each step in a sequence of Morse code. This unique pattern of activity in the hidden layer makes it possible to learn each item in a sequence in the appropriate order because the activation pattern is used on subsequent time-steps to learn the next item in the sequence, thus linking each element up in a sequential manner.

Learning the Morse code is actually quite a difficult problem for the model because of the large number of possible output sequences generated from only two output variables, i.e. 26 patterns of length one to four are represented using just dot and dash. Such a ratio means that the patterns will be harder to discriminate as they are very similar. The fact that the patterns vary in length also complicates the task, since the pattern for letter I (..) starts with exactly the same sequence as that of letter S

(...) which is also identical to the starting sequence of letter H (....) (A full encoding of the Morse code can be seen in Appendix I.). Thus there are many repeated sub-sequences and much overlap between sequences throughout the data set.

One way that the model overcomes this potential problem is to learn a unique cue for each letter. This is achieved on the first time-step during training when the model is taught to reproduce the input pattern on the output layer. This step ensures that each sequence can be uniquely identified. Because this initial mapping is unique, a dedicated hidden unit will carry that mapping and therefore be used in the next time-step of learning as part of the input. This procedure is repeated on subsequent time-steps, and therefore guarantees a unique pattern of input on each time-step. So regardless of the target pattern of activation on the output, a unique mapping is created for each item in a particular sequence.

Summary

The original theory of APECS, as described by McLaren (1993) has been successfully extended to a recurrent network topology that supports sequence learning. The REAPECS model performs well on the Morse Code problem when each letter of the alphabet (i.e. each input pattern in the data set) is represented by an individual node. However, the problem of interference, which was the original motivation for APECS has not been tested because the input coding scheme represents each letter as an independent entity. A more realistic distributed representation for coding the letters of the alphabet should not raise any problems for REAPECS. This is tested in the next simulation.

6.2.2 Model 6.2

Model 6.2 was designed to demonstrate that the theory of REAPECS also applies to different input coding schemes, namely one that employed a distributed

representation of input patterns. APECS was originally designed by McLaren (1993) to overcome the problem of catastrophic interference from similar input patterns, so there was no reason to suppose that employing a distributed input coding would present any problems for the model. The distributed representation of the alphabet was adopted, as previously documented in chapter 5. It was predicted that the same results would be obtained from Model 6.2 as were obtained from Model 6.1.

6.2.2.1 Architecture

The same basic architecture used in Model 6.1 was adapted to accommodate the new representation for letters of the alphabet described above. Each letter of the alphabet was represented as a 35-bit vector, which when read from left to right corresponded to a (top to bottom) list of rows from the 5x7 pixel grid (as in Model 5.3 of the Jordan sequential network described in chapter 5). Figure 6.5 illustrates the architecture of Model 6.2.

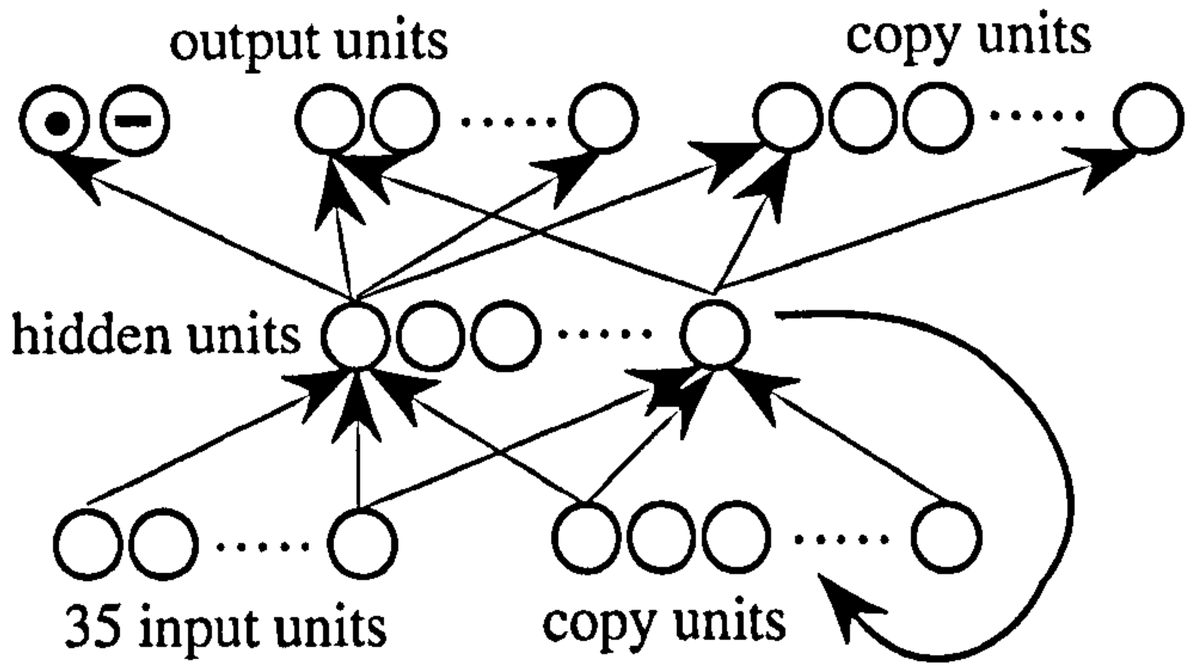

Figure *6.5.* The architecture for Model 6.2 is essentially the same as for Model 6.1 except that the external input units consist of 35 nodes which code each letter of the alphabet, distributed to reflect a 5x7 visual pixel grid.

6.2.2.2 Algorithm and Method

The same algorithm and method that were used in simulations of Model 6.1 were used again for the simulations of Model 6.2.

6.2.2.3 Results

The average MSE of ten simulations (each with different random weights) before training was 0.674787. This is higher than the average MSE of simulations of Model 6.1 before training and can be accounted for by the different input representations. For the first time-step in each sequence, the target output required many more output nodes to be on. As explained in the previous discussion section on Model 6.1, the default output values for untrained nodes is 0.5, hence if many more output nodes have a target greater than this value, the average MSE will be much greater.

After one pass of training on the data set none of the sequences were correctly recalled, although this is not surprising because the exposure to the data is so brief, and the results from Model 6.1 are similar in this respect. However, after several passes through the training data the results from Model 6.2 did not follow the pattern of results of Model 6.1 as was predicted. After Model 6.1 had had three passes through the training data all the sequences were correctly recalled (see Figure 6.4). After five passes through the training data with simulations of Model 6.2 only a few of the sequences were correctly recalled. Even after 20 passes through the training data, performance on Model 6.2 for both its MSE and the number of correctly recalled sequences did not significantly improve. It did not seem likely that further training would improve matters. These results are shown in Figure 6.6.

A closer look at the actual sequences output from Model 6.2 revealed some surprising patterns. The initial step in sequence learning using REAPECS involves learning to repeat the pattern of input on the output units so that a unique cue is learnt for each sequence. Inspection of this initial output pattern from the simulations showed that this was not being correctly learnt. Letter representations that shared similar or overlapping representations were interfering with each other, making

discrimination between similarly represented letters impossible. For example, letters such as 'U', 'V', 'W', and 'B', 'D', and 'N', 'M' were causing catastrophic interference between patterns. The pattern of learning the initial cues for each letter was not systematic across trials. For example, the correct pattern for letter 'A' may have been correctly learnt on one particular trial, but on subsequent trials training results in an incorrect pattern being learnt for letter 'A'. This is a classic sign of catastrophic interference as described earlier (McCloskey & Cohen, 1989).

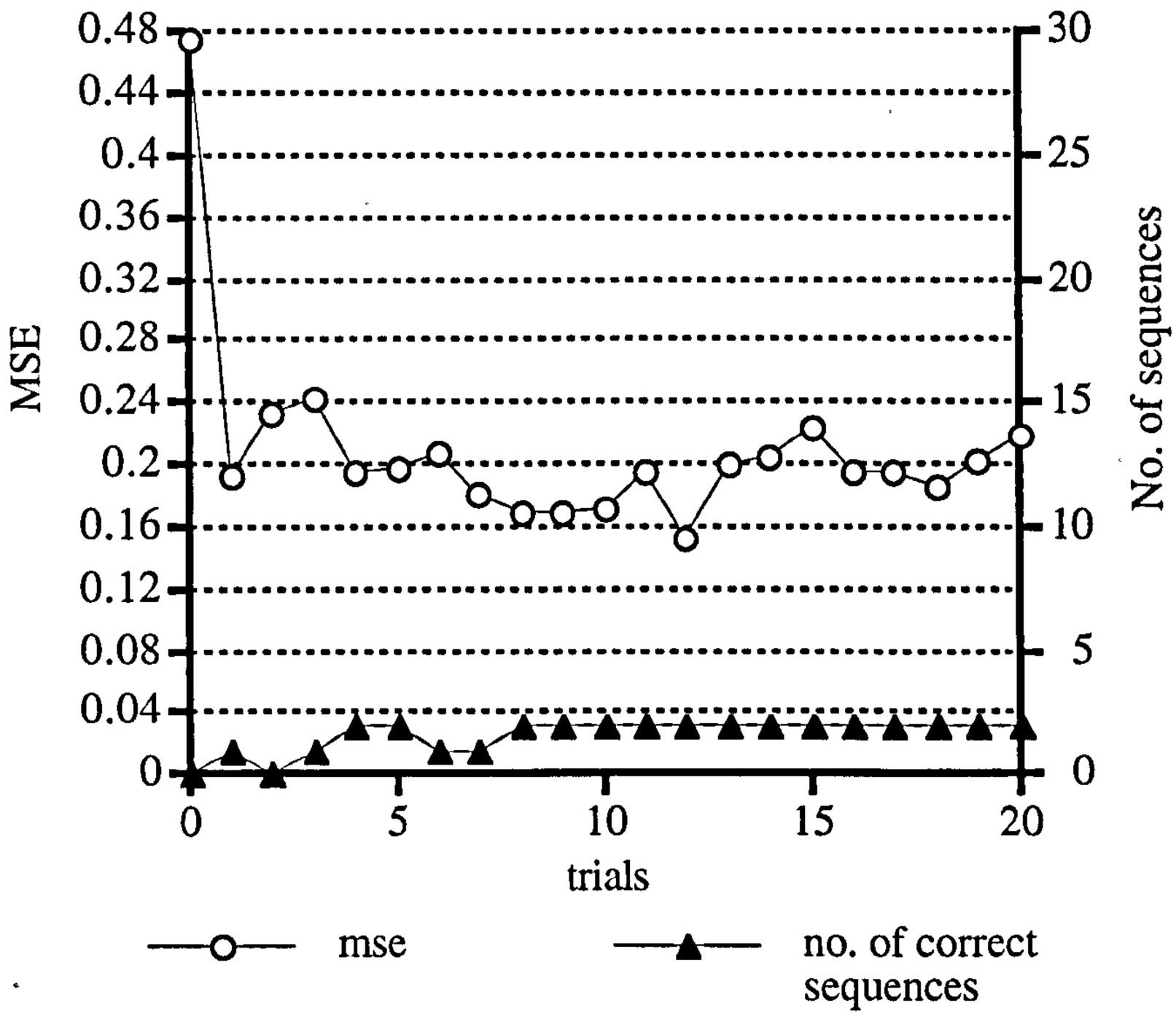

Figure *6.6.* A graph of the average performance of ten simulations of Model 6.2, each assessed and then averaged after the completion of each trial (of a total 20). The left ordinate plots the averaged MSE of the simulations, and the right ordinate plots the average number of correctly recalled sequences, against the number of successive trials as abscissa.

Other letter representations appeared to dominate learning, in that for example the representation for letter 'M' (or very similar) was being recalled for many other letters on some trials. These results indicate that catastrophic interference is not being prevented, and that mappings are not being protected once they have been learnt.

6.2.2.4 Discussion

This result was somewhat of a surprise because the central claim made by APECS is to prevent interference from similar input mappings having catastrophic effects upon previous learning. In other words, this sort of task is precisely what REAPECS should be good at. However, if the uniqueness criterion is not met for any one of the time steps of sequence learning, then the APECS algorithm makes no claims to its ability to distinguish between the associated sequences. This is obviously the case for Model 6.2.

The amount of training (i.e. the number of trials) the model receives might appear quite minimal; only twenty trials. However, learning each mapping involves 1000 cycles of learning per mapping per trial. Therefore after twenty trials through the training data, each mapping will in reality have received 20,000 iterations of learning. REAPECS also works in such a way that the weights in the network quickly develop large values to speed the time taken to find a solution, hence 20,000 iterations of training for a REAPECS model equates to a very substantial amount of training, by which time the model will be heavily biased towards its current solution, be it correct or otherwise. The model is therefore less than satisfactory.

6.2.3 Summary

Model 6.1 was a demonstration of how the APECS algorithm (McLaren 1993) could be adapted in the style of a recurrent network similar to that of Jordan (1986b) and Elman (1990) to form REAPECS. Learning from a simple (local) representation of the alphabet, Model 6.1 showed the ease with which it was able to learn the Morse code. The exact same problem had already been solved by a sequential network (Jordan, 1986b) in chapter 5 and Model 6.1 in this chapter acted merely as a demonstration that APECS could be modified to learn to recall sequences just like

Jordan's network, but without the end of target sequence ambiguities characteristic of Jordan's network.

Model 6.2 was designed to test the robustness of REAPECS across different input representation schemes. It was predicted that the central claim of APECS (a solution to catastrophic interference) would be upheld such that the results obtained would be very similar to those from Model 6.1. However, this was not the case and Model 6.2 suffered from interference characteristic of the simple feed-forward networks reported by McCloskey and Cohen (1989). However, although the results were not as expected, Model 6.2 was able to learn to recall some of the Morse sequences, only confusing the letters that were similar in appearance and performing even better on the letters that were more distinct. The results from Model 6.2 suggest that if the mechanisms that prevent catastrophic interference, already in place in REAPECS, could be refined, then a more reliable model might be produced.

The mechanisms at work within REAPECS are not particularly transparent, which makes it hard to tell with any certainty the exact locus of the problem that gives rise to these results. This is a characteristic of many connectionist models. The results obtained from the simulations of Model 6.2 therefore prompt a much more detailed investigation into the mechanisms at work within RAPECS before it can be applied to speech production.

6.3 The Extended REAPECS Model

The aim of this section is to find a solution to learning the Morse Code problem from distributed input representations using the REAPECS model. The results from Model 6.2 falsified the prediction that REAPECS would be robust over different input representations. However, REAPECS is a system of mechanisms that must operate in a complementary fashion to achieve the desired effect of learning. The implementations of these mechanisms in Model 6.1 which were used as the basis for

Model 6.2 may not necessarily be optimal. It is possible that different realisations of these mechanisms within REAPECS would yield a superior implementation, capable of learning from both local and distributed representations.

In the next section, I first explain the functional relevance of the most important mechanisms affecting performance and then I describe some alternative settings or ways of realising them. Another simulation follows, demonstrating the effects of the alternative implementations.

6.3.1 Modifications to the REAPECS Model

6.3.1.1 Similarity of the plan inputs

The external input to the Model is constrained to be a pattern of positive activation values on the input units, specifically either 0 or 1. The *input range* constraint could alternatively be extended to include negative activations (i.e. a form of inhibition) such that the activation values could be either -1 to +1. This would provide greater discrimination between similar input patterns and should help the model to distinguish between patterns faster than with a more constrained range.

6.3.1.2 Serial order links

The serial order mechanism operates by creating links from one time-step in a sequence to the next. This is controlled by the way the activation levels of the copy back units are derived from the hidden units. Their precise value is restricted by the *copy-back* constraint to a derivation from the activation levels of the hidden units on the previous time-step. They can be derived in several ways.

(i) A straight copy of all hidden units. This is the least stringent setting for the constraint, where the exact activation values of the hidden units are copied without modification to the copy units.

(ii) Copy selected hidden unit only. If some internal mechanism could remember which hidden unit had been selected to learn the mapping in question, then this hidden unit could be used to set its corresponding copy unit to full activation, whilst all others would remain set at zero. This is the most stringent evaluation of the constraint and also the most unlikely.

(iii) Copy only the most active hidden unit. If the copy-back mechanism was made competitive then only the unit with the greatest activation would be copied (with an activation of 0.9).

(iv) Copy all hidden units which fulfil the criteria for a unit to be declared 'on'. A thresholding procedure could be carried out on the hidden units so that all units with activation values above 0.6 were (the units that were effectively 'on') copied to the copy units, while all hidden units whose activations fell below 0.6 were copied as zero. This is appealing because the relative activations of the more active units would be carried forward to the next element to be learnt, also allowing for a slightly distributed representation to develop at the hidden layer. This would in turn allow for some shared representation throughout the sequence to develop on the hidden units. This is the method that had been used up to now.

6.3.1.3 Sequence protection

The bias of all hidden units is modified by the *sequence protection* constraint so as to prevent inappropriate units from interfering with selected ones. Even if a hidden unit was not selected its bias can still change. The bias modifications are determined by the sign of the raw error score of the hidden unit and the selection status of that unit (i.e. whether or not it has been selected to mediate the mapping).

If the raw error score of a hidden unit is negative, then its bias is decreased in proportion to its overall error score. If its raw error score is positive its bias is

incremented slightly unless the unit has been selected to mediate the mapping in which case its bias is unaltered. Hence inappropriately active hidden units develop a large negative bias to prevent them becoming activated in future when that input is presented. Protection can be further enhanced if the constraint takes into account the current activation level of each hidden unit as the bias is adjusted. This would affect the modification of biases to units whose suitability score was greater than zero, but had not been selected to mediate the mapping. This method is most useful when the veto parameter takes a large negative value because it provides an extra mechanism to control hidden unit development when more units can compete for the mapping. Making the change in bias also proportional to the activation of these units would decrease the bias of those units with a higher activation more than those with a lower activation, and thus provide more protection for the selected unit to mediate the mapping at hand.

6.3.2 Model 6.3

Model 6.3 was designed to explore how the above mechanisms affect the performance of the REAPECS model. The same task, that is the Morse code problem, with a distributed input representation of each letter of the alphabet was used again.

6.3.2.1 Architecture

The same architecture was used for Model 6.3 that was used for Model 6.2.

6.3.2.2 Algorithm and Method

The same method and algorithm that were used for simulations of Model 6.2 were also used here. However, alternative implementations of the *input range* and *sequence protection* constraints were explored. The input range constraint now allowed activation of the input units to be either -1 or +1. The sequence protection

constraint was implemented such that the biases of the hidden units (their protection mechanism) are changed also in proportion to their activation level. The *copy-back* constraint was implemented such that during learning only units that met the criteria to be deemed 'on' were copied back, but during recall the actual activation pattern of the whole layer was copied without modification.

Simulations were run started with random weights in the range -0.02 to +0.02. Two versions of Model 6.3 were simulated, one with the input range set to the between 0 and 1 constraint (Model 6.3.1), the other set to the between -1 and +1 constraint (Model 6.3.2). Performance was assessed in the same way as for Model 6.2 in terms of the MSE score.

6.3.2.3 Results

Figure 6.7 shows the rate of change of MSE of the two models during training. The best results are clearly those obtained from Model 6.3.2 as 100% of the data set was successfully recalled and it had the lowest MSE score. Model 6.3.1, although having a low MSE score, was able to recall just 26.9% of the data set correctly.

The better result was obtained by implementing the constraint on the input range such that both positive and negative values are allowed (Model 6.3.2) and by implementing the constraint on the protection mechanism such that the change in bias of hidden units was also proportional to the activation of that unit. This implementation has the effect of making each letter representation easier to learn in the first instance, therefore removing the problem of pattern interference and is reinforced by giving a higher level of protection to each mapping within each sequence.

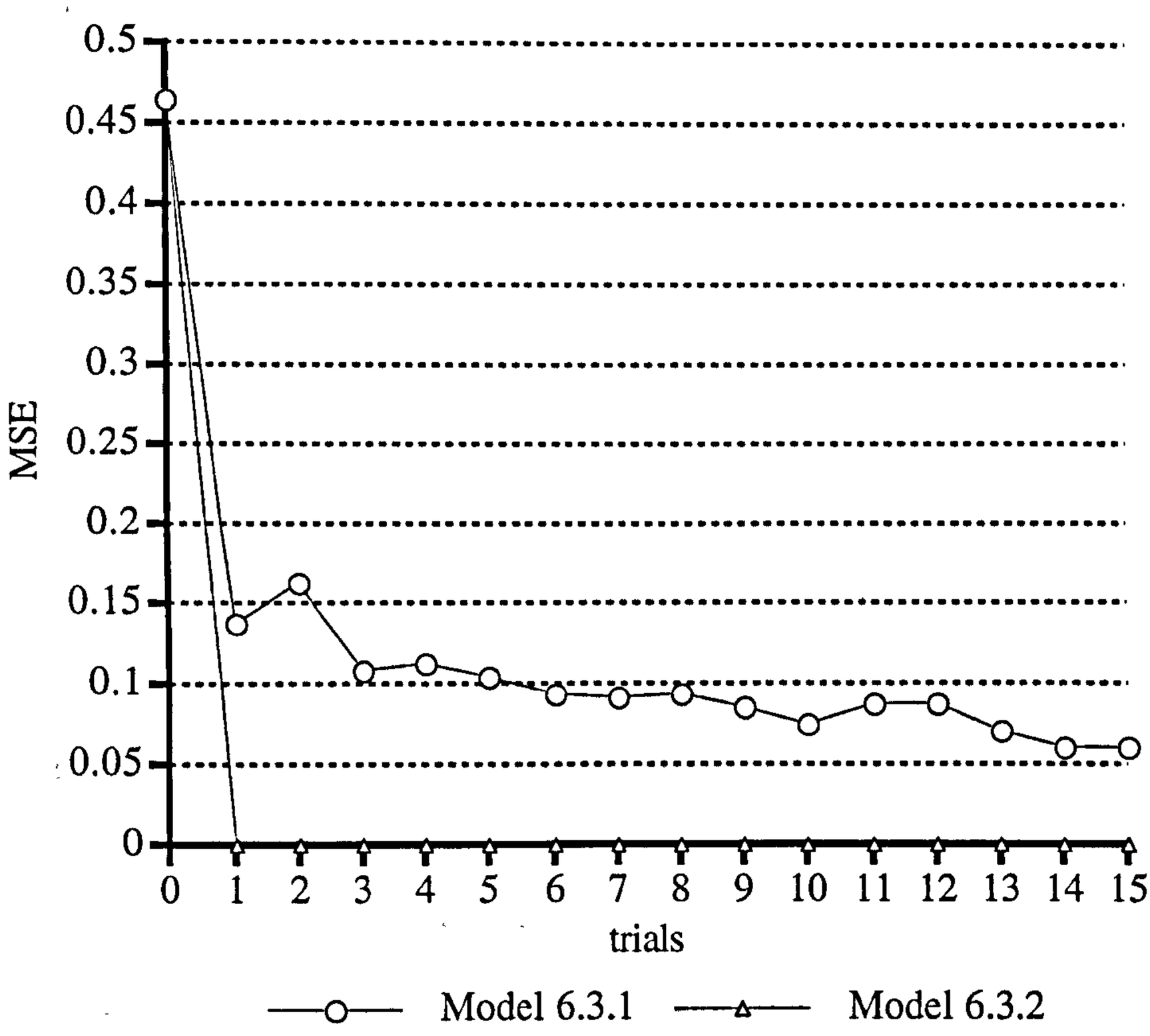

Figure 6.7. The rate of change in MSE of Model 6.3.1 (input range set to the between 0 and 1 constraint) and Model 6.3.2 (input range set to the between +1 and -1 constraint) during learning the Morse code problem.

6.3.2.4 Discussion

The results show the impact of varying two constraints in the REAPECS model. The best performance from the model is obtained when the input range mechanism allows both positive and negative values, rather than constrained to strictly non-negative values. The mechanism affecting the discrimination between letter representations seems to be the most important mechanism when working with distributed representations. Thus when the model is more able to discriminate between the letter cues which represent each sequence, and greater protection is given to their learning, then the recall of the corresponding Morse sequences is performed to perfection. It would seem from this interpretation that the problems encountered by Model 6.2 were due to the inability of the REAPECS model to represent uniquely the

cue for each letter. Once a unique representation had been learnt for each sequence, then the actual learning and recall of the elements of the sequence was achieved. This is in accordance with the fact that a model using a local representation of letters and hence no potential interference between letter patterns had already been successfully demonstrated with Model 6.1. Model 6.3 has shown how certain modifications to the REAPECS model can yield a more robust model of sequence learning. Each sequence is uniquely represented such that learning to associate similar cues with different sequences is performed to perfection.

A further important aspect of performance usually associated with connectionist models is generalisation. This can be briefly described as the ability of a model to respond to an unseen (and hence untrained) input cue in a manner that reflects the experience it has acquired from the data on which it has been trained. In other words, it should be able to extrapolate a suitable response from similar patterns already learnt. A model can only provide a generalised response if the form of its response is in a distributed format as well as its input. Although REAPECS is robust in the sense it can still recall sequences when given incomplete or noisy letter cues, REAPECS will not generalise from unseen input cues for the reason that the output is described locally. The only extent to which the model can generalise is to produce the sequence which is associated with the input cue that most closely matches the unknown input. This may be described better as robustness and certainly is more desirable than generalisation in the more traditional sense where sequential information is concerned.

The modified REAPECS Model 6.3.2 out performs the Jordan network in terms of reliability to reach a solution, the REAPECS model was also able to terminate production at the end of each sequence, and did not experience the problem of falling into a never ending repetition of post-end-of-sequence elements.

6.4 Summary

The REAPECS model of sequence production was motivated on the one hand by the limitations of recurrent neural networks and on the other by salient features of McLaren's (1993) APECS model. The aim of the model was to develop a basic model of sequence production that would be suitable as a model of serial order in speech production, yet overcome the limitations of Jordan's sequential network. These include problems with limits on the lengths of sequences that can be learned (Servan-Schreiber et al., 1991) and problems with terminating sequences. Human processing appears to handle these problems relatively effortlessly.

A successful model of sequence production has been demonstrated by the REAPECS model by learning the Morse code problem, in which the appropriate sequences of Morse code are learned for each and every letter of the alphabet. The Morse code shares some similarity with speech production in that many sequences can be formed from a limited set of elements by combining and repeating them in a different order. Although several modifications of the original model were necessary to find a working model, a more reliable model has been produced which can learn a variety of variable length sequences from similar but different input cues. This characteristic of REAPECS is of interest in the study of speech production because many phonetic sequences must be learnt from similar lexical and semantic cues in order that our thoughts be articulated into speech. The finding that certain phoneme errors (anticipations, perseverations, exchanges) are constrained by distance (Garrett, 1975) suggests that the locus of these errors is particular to the representation of the sequence within these distal constraints. REAPECS also constrains the representation of particular sequences by using a unique set of hidden units for particular sequences. It is now of interest to see whether these representations fall prey to movement errors as observed in corpora of spontaneous speech errors made by humans. Now that the basic properties of sequence production have been captured by REAPECS, a more

relevant test of its interest to speech production and phoneme errors can be carried out by applying the model to a task of learning to map semantic and lexical representations to phoneme sequences. This is the topic of the next section.

6.5 The REAPECS Model of Speech Production

It is desirable when modelling speech production that the model should be analogous to the particular part of the speech process being modelled. Speaking begins with an intention to convey a message. The representation of a message is in itself is not a trivial matter, but it can be viewed as a pattern of activity of the semantic information within the message (e.g. Hinton & Shallice, 1991; Harley & MacAndrew, 1992). How such an intention is invoked and the exact form of the message is beyond the scope of this thesis, but a possible representation of the semantic information contained within the message is presented in this section as a means of describing the input to the model. A phonetic representation is easily described in terms of distinctive features as the output of the model. In learning to produce sequences of phonemes, each identified by a unique semantic representation, it is hoped that subsequent lesioning of specific sites within the model will lead to the production of serial order phoneme errors. It is predicted that by lesioning the serial order mechanism within the model, the output sequence of phonemes will be subsequently misordered.

6.5.1 The Word Set

The words to be used as the data set for the model were chosen from the Oxford Psycholinguistic Database (Quinlan, 1992). They were chosen with two considerations in mind. First, the computational constraints concerning the number of words that could be learnt in a reasonable amount of time were taken into account, and second, the way in which the words were to be represented by the model was also

considered. The latter consideration meant that words that could easily (relatively) be visualised and hence described by semantic features should take preference to those which could not. For example, words such as "duck", "tent" and "frog" are all easy to picture and therefore describe. However, words like "love" and "fate" are hard to imagine visually and cannot be easily described in terms of features. A set of words was finally selected from the Oxford Psycholinguistic Database that met the following criteria. Each word must be a monosyllabic noun and could contain no more than four letters or nineteen phonemes. Each word must have a Kucera-Frances (1967) frequency of no more than 20, a Familiarity score between 450 and 700, a Concreteness score between 550 and 700, and an Imageability score between 550 and 700. The Familiarity, Imageability and Concreteness scores were calculated by Quinlan (1992) by blending together the familiarity norms of Gilhooly and Logie (1980), Paivio (unpublished, as cited in Quinlan, 1992) and Toglia and Battig (1978). (A full description of the meaning of each criterion can be found in Quinlan, 1992.) These criteria were selected so that the set of words they define would be easy to describe by semantic features, and also to restrict the number of words in the set to reduce the computational load. This yielded 87 words.

6.5.2 Model 6.4

The input layer of the model must reflect the semantic content of the word to be produced. Rather than dedicating individual units to entire concepts, or words, which would represent a localist coding scheme, a distributed representation of each concept or word was favoured. This coding scheme involves assigning individual units to single semantic features. For example one unit may represent the semantic feature *small*, another *plant*, and so on. The input vector, then, would represent a collection of single features that together convey the meaning of a particular word. The set of semantic features used for the purposes of Model 6.4 was selected from Harley

(1993b) and Hinton and Shallice (1991). A few extra were chosen specifically to better describe some of the words within the set chosen from the Oxford Psycholinguistic Database (Quinlan, 1992). The total number of features totalled 36 and are listed in Table 6.1.

Table 6.1.

Semantic feature descriptors, based on Harley (1993b) and Hinton and Shallice (1991) used to define the words in the word set selected for Model 6.4.

Semantic feature descriptors			
animate	small	instrument	wood
plant	mobile	male	liquid
man-made	domestic	female	external
flies	has-components	old	internal
4-legs	aqua	young	human
2-legs	edible	component	living
0-legs	carnivore	hard	clothing
big	herbivore	soft	glass
medium	omnivore	metal	round

Each feature was designed not to be too specific, as it may be used in the description of many words. For example the feature *instrument* applies not only to words such as "bell" or "drum" (which are musical instruments), but also to words like "fork", which is an instrument for eating. Thus features are used in their most general sense to describe concepts. Each feature in the vector was either assigned a value 1 (if the feature was true of the word) or 0 otherwise. A full listing of each word and its corresponding semantic feature description can be found in Appendix II.

The output layer of the model must represent the corresponding output of the human process being modelled, in this case a sequence of phonemes. Phonemes can be described in a standard way by a set of distinctive features. These features describe the activities and shape of the vocal apparatus necessary to produce a particular sound. Describing phonemes in this way has the advantage that the similarity between certain phonemes e.g. /p/ and /b/ is easily captured. A distributed representation was therefore chosen for the output of the model, each unit representing a different distinctive feature of articulation. A pattern of activation

over these units corresponds to a particular phoneme. A set of features was proposed by Chomsky and Halle (1968) which has been adopted by many as a working standard. In this system, the distinctive features are binary, that is they can only have one of two possible values, + or -. Variations to the Chomsky and Halle system have developed (e.g. Lass, 1984), which has inevitably lead to a number of classification systems that are not always consistent (Singh, 1976). The distinctive features I have chosen are a modified version taken from Sloat, Henderson-Taylor and Hoard (1978) who in turn based their set on that of Chomsky and Halle (1968). The distinctive features used in the following simulations are summarised in Table 6.2.

Table 6.2.

Definitions of distinctive features used in the output representation of the REAPECS speech model.

feature name	definition
consonantal	produced with contact between articulator and articulation
sonorant	vocal tract shaped so air flows unimpeded through nasal or oral cavity
syllabic	segments with greatest prominence within a syllable
high	tongue body above a neutral position
low	tongue body below a neutral position
back	retraction of tongue from neutral position
front	no retraction of the tongue
rounded	rounding of the lips
interrupted	a complete blockage of the airstream during part of an articulation
strident	vocal tract shaped so air only flows through a narrow gap in the centre
nasal	some or all of the air is expelled through the nose
lateral	airstream is diverted laterally around the tongue
voiced	periodic vibrations of the vocal cords
tense	relatively high tension in the oral cavity muscles
coronal	front or apex of tongue is raised to form a total or partial obstruction
anterior	sounds made at or in front of the alveolar ridge

The symbol notation for the phonemes was based on the International Phonetic Alphabet notation, except for those phonemes which were represented by special hieroglyphic characters. These have been represented with alternative characters found on the QWERTY keyboard for the convenience of computer modelling. The

actual list of phonetic symbols and an example of their usage within English Received Pronunciation (RP) can be found in Appendix III, and their featural specification is given in Appendix IV. For completeness, Appendix V contains the list of words to be used in the simulations together with their phonetic representation (i.e. the target output of the model).

Phonemes can be split into two classes; consonants and vowels. While the duration of consonants varies little, vowels vary much more. Vowel sounds are not absolutely pure in quality, but can de divided into two categories - monophthongs and diphthongs, according to quality. The so called pure vowels [i, I, e, @, ^, a, o, c, U, u, 3], or monophthongs, can also be divided into long [i, a, c, u] and short [I, e, @, ^, o, U, 3] vowels. A diphthong [eI, aI, cI, 3U, aU, I3, e3, U3] is a sound that forms a glide within a syllable. That is, the quality of the produced sound does not remain constant (unlike a monophthong). Diphthongs may be described as being composed of a starting (or first) element and a second element or point to which the glide is made (e.g. Gimson, 1980). For simplicity, both long and short monophthongs and diphthongs are treated in the same way by the model.

The first aim of Model 6.4 is to learn to produce the correctly ordered sequence of phonemes when presented with the corresponding semantic representation of that word for all words in the training set. A reduced set of 50 words was chosen from the original set of 87 selected from the Oxford Psycholinguistic Database (Quinlan, 1992) to shorten the training time. The second aim is to compare the models errors with the speech errors made by adults after the serial order mechanism has been lesioned.

6.5.2.1 Architecture

The architecture of the speech production model is based on the general REAPECS architecture as described in Model 6.3. Figure 6.8 shows the model and its connectivity.

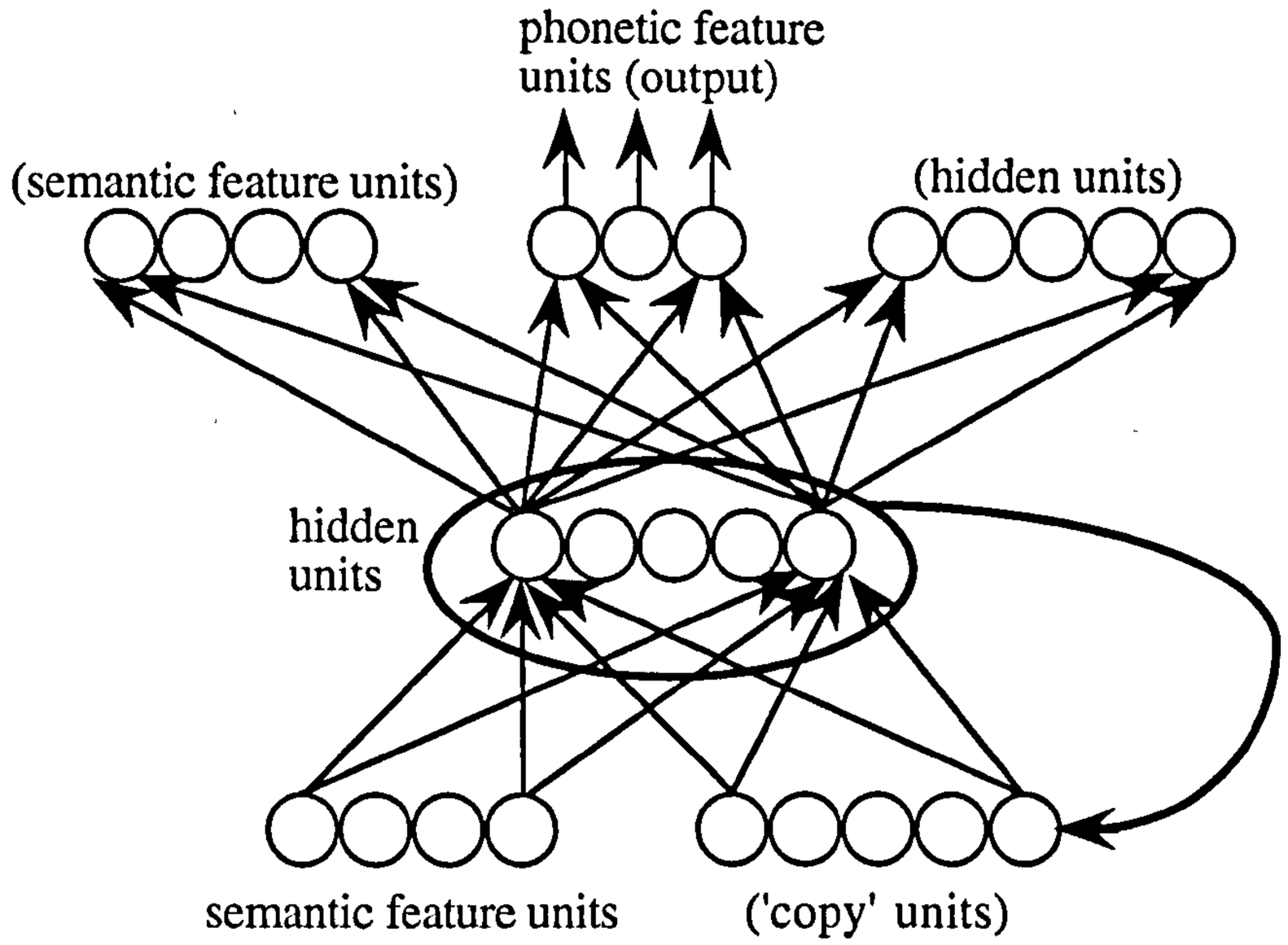

Figure 6.8. Architecture of Model 6.4. Not all units or connections have been shown for the sake of clarity, although the figure is an accurate representation of the general architecture.

The input layer consisted of 36 units in one cluster which represented the semantic feature descriptors previously described. The other cluster of units in the input layer consisted of 500 copy units. The hidden layer consisted of 500 hidden units; this number was based on the total number of phonemes in all sequences of the word set. The output layer consisted of three clusters of units. The first cluster contained 36 semantic feature units, the second contained 16 phonetic feature units (as previously described), and the last cluster contained 500 copy units, to help identify uniquely each element within a sequence.

6.5.2.2 Algorithm and Method

The same algorithm and method that were used in simulations of Model 6.3.2 were used again for the simulations of Model 6.4. The output of the network was a pattern of activation across 16 distinctive feature units. This can be thought of as describing a point in articulatory space. In order to assess which particular phoneme

was represented at each time step, the pattern of activation on the output units was compared to the activation patterns for all the phonemes used in training (i.e. all English phonemes). This was achieved by defining a proximity measure (Hinton & Shallice, 1991) of the output vector to each stored phoneme by measuring the cosine of the angle between the actual output vector and all the stored phonemes. The stored phoneme that was subsequently output by the model was the one with which the actual output vector had the smallest angle.

Once the model had been trained to maximum performance, lesioning was performed independently, in two ways. The first method involved adding uniformly distributed random noise directly to the activations of the hidden units. The second method involved the addition of uniformly distributed noise to the weights connecting the copy back units and the hidden layer. These weights were chosen because they represent the sequential mechanism in the trained model.

6.5.2.3 Results

Three simulations of the model were run where each was initialised with a different set of random weights. The results from all three simulations were very similar. The results from the simulation that performed best in recalling the word set after training are given here. After two passes of training on the data set, the model was able to satisfactorily recall 47 out of the 50 words. Further training on the data set did not improve performance above 47 correct. The best simulation of the model therefore gave a correct response for 94% of the word set.

Lesioning the model resulted in an unexpected type of error, in which the general ordering was preserved but the sequences were truncated. Table 6.3 below shows the results after the first lesions, which were made by adding noise to the hidden unit activation levels. The results of the second lesion, to the weights

connecting the copy-back and hidden units are presented in Table 6.4. The noise SD indicates the standard deviation from zero of the noise added to each unit, or weight.

In Table 6.3, for uniform noise with an SD of up to 0.75, all responses were classed either as correct, no response (no output at all) or by the initial number of correct phonemes produced (one, two or three) before the sequence was truncated, for each word in the trained set. Figures are given as percentages of all responses from the complete word set.

Table 6.3.

Performance of Model 6.4, as a percentage of the total number of sequences in the training set, after lesions to the hidden unit activation levels.

noise SD	correct sequence	no response	number of phonemes correctly recalled		
			1st	1st two	1st three
0	94	6	0	0	0
0.02	92.4	4	2.4	0	0.8
0.05	65.6	4.8	22.8	6.4	0.4
0.1	57.2	26.4	13.6	2	0.8
0.2	51.6	41.2	6	1.2	0
0.25	45.6	50	3.2	0.8	0.4
0.5	52.8	45.2	2	0	0
0.6	52	46.4	2.4	0	0
0.75	40	57	2.4	0	0.4

When the SD of noise added to the hidden unit activation levels was increased above 0.75, a somewhat random noisy response was observed, with 7% correct responses, 4% no response, 39% of incorrect length and incorrect phonemes, 9% word substitutions and 41% phoneme substitutions.

Table 6.4.

Results from lesioning the weights connecting the copy back and hidden units, in percent, in a fully trained simulation of Model 6.4.

noise SD	correct	no response	number of phonemes correctly recalled: 1st two	1st three	1st phonemes at end	misc.
1.25	94	6	0	0	0	0
2.5	72	4	12	10	2	0
5	0	4	40	16	0	40

In Table 6.4, no effect on performance was observed until the SD of the random noise was greater than 1.25. Responses were classified the same as for the results from lesions to the hidden unit activations, except for an extra category in which the correct sequence was produced but additional miscellaneous phonemes were also produced immediately after the last phoneme in the target sequence. This is shown as the final column in Table 6.3, above. As was the case for the lesions to the hidden unit activations, the performance declined as the SD of the noise increased.

6.5.2.4 Discussion

The results were somewhat disappointing in that none of the phoneme movement errors found in the human data were produced by lesioning the model. The addition of random noise to the activation of the hidden units or the connecting weights, produced a type of deletion error in which sequences were truncated (i.e. the first one or two phonemes were produced, but the rest were omitted). Deletion errors do occur in human speech, but not of the same type which was produced by the model. In human speech, phonemes are sometimes omitted, as in [6.1],

[6.1] whiskey -> whi_key

but the other phonemes in the sequence are unaffected and still spoken. When phonemes are lost from the ends of words (in human speech), it is often in the context of a more complex error where the beginning of the next word is also omitted and the two words blend, for example as in [6.2].

[6.2] I've been interviewing **matudents** all morning -> ma**ture st**udents

The model produces truncation errors when the noise produced by either lesion masks the information on the hidden units that are involved in the recall of the target sequence. The way in which the model fails is not appealing in that graceful degradation is not displayed; rather an all or nothing type behaviour is observed. This is due to the somewhat rigid and local way in which phonemes are chained together with one large weight per link.

The only sort of error that was produced which is also observed in the human data were word (as in [6.3]) and phoneme substitution errors (as in [6.4]),

[6.3] **beak -> lamp**

[6.4] **beak -> geak**

where the word "beak" has been replaced with "lamp" and in the latter case where the phoneme /b/ has been substituted by /g/. However this type of error does not offer any insight into the serial order of phonemes in speech, because there is no movement of phonemes within the sequence.

The phoneme substitution errors produced by the model can easily be accounted for by the distributed representation of phonemes on the output layer of units and the subsequent addition of random noise impinging on these units. As either the connection strengths become more noisy or too many hidden units become highly activated, the more noisy the output of the phonetic feature units becomes and the less closely the phonetic feature units match the target phoneme. Thus phonemes that are similar to the target will also become highly activated, eventually resulting in one of them providing a better match to the actual output than the original target, leading to a

substitution error. Hence in the case where sufficient noise was added to the hidden unit activation levels a large proportion of the responses produced were substitution errors. The fact that 39% of responses in this condition were characterised by sequences containing the wrong number of (mainly) wrong phonemes can also be explained in a similar way. That many of these responses contained the wrong number of phonemes is due to the way in which serial order is achieved in the model. That is by copying the hidden unit activation levels back to the hidden layer at the end of each time-step. If sufficient noise is present during recall then many more sequences become active which in turn means that alternative sequences may also be recalled. If a longer sequence than the target is also activated then too many phonemes will be produced. Obviously, the production of miscellaneous phonemes at the end of words is not at all realistic.

6.6 General Discussion

Collectively the results suggest that the model is little more than a novel method of implementing a table look-up mechanism in which a rigid one-to-one mapping is enforced between each word and its associated phoneme sequence. Presentation of the semantic representation of a word results in its associated sequence of phonemes being effectively looked up by sequentially activating the hidden unit with which each phoneme has been associated. For example, Figure 6.9 illustrates a typical learnt representation for the production of the word "cat".

At t = 0 presentation of "cat" activates the second hidden unit which effectively looks up the stored representation of the first link in the sequence for "cat". At t = 1, this hidden unit is used to activate the next in the link to produce the first phoneme in the sequence, /k/. The procedure is the same for the rest of the sequence.

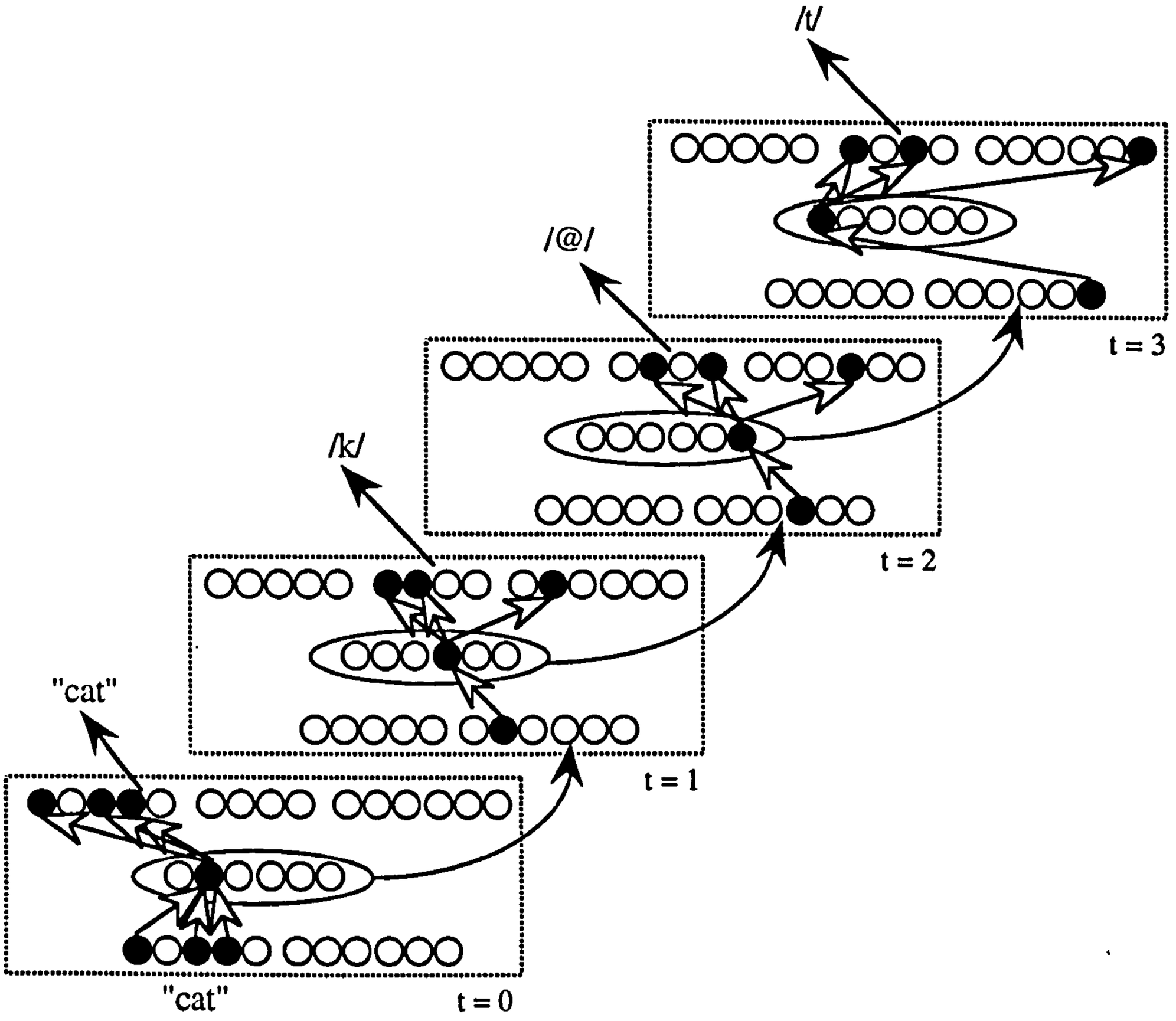

Figure *6.9.* Production of the word "cat", broken down into time steps in the REAPECS speech production model. Filled circles represent highly active units and lines indicate facilitatory weights.

Notice that the same hidden unit is never used for more than one mapping. This is one of the central mechanisms that allows REAPECS to overcome the effects of catastrophic interference. Even identical phonemes, such as the /@/ in /k@t/ and /t@p/ will not share the same hidden unit representation. In this respect, the /@/ in /k@t/ and the /@/ in /r@t/ are as different from each other as any other phoneme. Thus each word is represented in the model as a chain of local hidden unit activity at each time step, in which case a set of conditional rules would also suffice (e.g. IF "cat" THEN /k/->/@/-> /t/ etc.). This is analogous to a table look-up mechanism.

In this light, REAPECS is similar in nature to Wickelgren's (1969) context-sensitive associative chain theory of serial order, where each element in a sequence is associated to its immediate neighbours. REAPECS mimics this theory by establishing a completely local link via the hidden layer between phonemes from the beginning to the end of the word. i.e. "cat" -> /k/ -> /@/ -> /t/. A separate chain would be generated for the word "tap" -> /t/ -> /@/ -> /p/. The chain is represented in REAPECS by two sets of connections. The connections between the copy back and hidden units represent the links between each phoneme in the sequence. The weights connecting the hidden units to the phonetic feature level represent the particular phoneme associated with each position in the sequence. REAPECS is thus able to produce sequences with repeated phonemes in much the same way as Wickelgren by using a token to represent each instance of each phoneme rather than a single type for each phoneme.

Such a theory of serial order has difficulties in accounting for certain serial order speech errors, as I have already argued in chapter 3. Although the model predicts substitutions, it is not likely that it could produce anticipations, perseverations or exchange errors. Non-contextual substitutions arise in the REAPECS model as a result of the similarity of featural representations of phonemes. Anticipations, perseverations and exchange errors on the other hand are not accounted for within the model because sequences are represented by linear links in the REAPECS model which serve only to activate the next element within the sequence. Therefore information regarding upcoming or past phonemes is never available concurrently with the production of the target phoneme. Additions and deletion errors are equally troublesome to explain, for the same reasons that associative chaining theories suffer, as I argued in chapter 3. It is not clear how the model could produce either addition or deletion errors or any errors that involve the movement of phonemes because the model represents each word as a series of unidirectional links

from one phoneme to the next in much the same way as associative chaining theories do. The REAPECS model is therefore subject to the same limitations as Wickelgren's (1969) theory of serial order.

6.7 Conclusion

Although the REAPECS model is indeed able to learn phoneme sequences from a semantic description of the word, it is also rather limited in its ability to provide an account of the serial order errors typically found in spontaneous speech production, that is phoneme movement errors such as anticipations, perseverations and exchanges. The REAPECS model predicts certain non-movement speech errors, such as non-contextual substitutions, and even blends of words and whole word substitutions are feasibly accounted for. However, the speech errors that involve the serial ordering mechanism, namely anticipations, perseverations and phoneme exchanges cannot be readily predicted by the current model. It is these errors that are of particular interest to the study of serial ordering in speech production. Thus although REAPECS provides a working model of serial order, it does not seem to be likely to be able to account for known serial ordering errors within the speech production literature. The fact that it fails to produce errors that fall into any of these categories means that it is limited in making a contribution to our understanding of how these types of error may occur, or even suggest a plausible account of a general theory of serial order for speech production.

The REAPECS model if nothing else demonstrates the limitations of associative chaining theories of serial order. In its most basic form, REAPECS is nothing more than a variation from the general topology of recurrent networks. Hence the recurrent network approach is also a class of associative chaining theories and REAPECS is simply a stronger model of associative chaining. Recurrent networks must also develop a unique pattern of activation to represent each item in a sequence (like

REAPECS) in order that different combinations of the same items and repetitions are possible. They differ from REAPECS in that these representations are allowed to overlap so that there is similarity between the links between sequences and generalisation to the next most likely item in the sequence being produced becomes possible. Hence they are weak models of associative chaining because the pattern of activation on the hidden units still serves to represent links in a chain which activate the next item in the sequence. Thus recurrent models, although capable of extracting statistical regularities in terms of phonetic regularity, will still face the problems of associative chaining theories in general. That is they are still primarily predictive of the next state only (given a past history of events, or links); and future (or past) elements are not actively represented at the same time as the current element, which is how anticipatory (and perseveratory) errors are revealed. In the same way that associative chaining theories fail to capture certain speech error phenomena, movement errors cannot be accounted for within the framework of recurrent models either.

The main conclusion to be drawn from all the simulations so far is that a model of speech production capable of predicting movement errors must do so by simultaneously activating remote as well as adjacent phonemes that are part of the planned piece of speech, an observation of Lashley (1951) which still appears valid. On the grounds of this conclusion, attention is now turned to a class of models where the simultaneous activation of planned output actions is indeed possible. The models described in the next chapter are competitive queuing (CQ) models, which use a control signal for the dynamic generation of serial order.

Chapter 7

7. Competitive Queuing Models

It has become apparent that models that do not allow the simultaneous representation of output information cannot provide an adequate account of certain sound movement error data. In this chapter I introduce a class of model, known as *competitive queuing* (CQ) models (Houghton, 1990; but see also Burgess & Hitch, 1992; Rumelhart & Norman, 1982), which address the problem of simultaneously representing output information and resolving its serial order. Houghton's (1990) CQ model serves as an introduction into a more advanced type of CQ model, the **OSC**illator-based **A**ssociative **R**ecall model. The OSCAR model is explained in detail, and simulations are described which show how the model linearises a simultaneously active sequence of phonemes into their correct order. The model also demonstrates the importance of the temporal constraint imposed upon the production of sequences.

7.1. Houghton's (1990) CQ Model

CQ models focus on the issue of producing serial order from highly parallel systems, although they are a different type of connectionist model compared with recurrent networks. The advantages of CQ models over associative chaining or recurrent type models have been demonstrated in fields such as speech production (Houghton, 1990; Houghton & Hartley, 1995; Hartley & Houghton, 1994) and spelling (Houghton, Glasspool & Shallice, 1994). Instead of producing each

phoneme on discrete time-steps or assigning them to slots in linguistic frames, CQ models operate by partially activating all the phonemes to some degree such that the most highly activated phoneme at any given time is the one which is output. This is achieved by using a dynamic *control signal* to maintain the correct activation gradient and a competitive filter that contains lateral inhibitory connections such that the most active output suppresses the others.

In Houghton's (1990) model of speech production, a CQ model is trained to produce serially ordered phonemes. Simple processing units, or groups of units, are arranged into a layered system like a feed-forward architecture, but the way in which they are connected is somewhat different. The properties of some of the connections also differ in that some units are connected by explicit inhibitory links. Units are grouped into three layers. There is a layer to represent the sequence information for each word: this consists of two nodes per word, one for the beginning and one for the end of each word, forming a pair. The next layer represents each phoneme, and the last layer is the competitive filter. Figure 7.1 shows the architecture of Houghton's model.

The word-pair nodes represent timing information for the beginning and end (or 'temporal edges') of each word. The dynamic co-activation of each word-pair node produces a time-varying control signal for that word, which is associated on discrete time-steps, by simple Hebbian learning (Rumelhart & McClelland, 1986), to all of the phonemes in the sequence being recalled. The control signal changes state over time and hence different states of the signal are associated to different parts in the sequence.

The control signal is also temporally correlated such that adjacent states of the signal are more similar than non-adjacent states. Thus all items in the sequence become activated to some extent (depending on the state of the control signal) simultaneously during recall. Their activation levels form a gradient such that when

sorted into descending order of activation, their correct sequential order is generally preserved. This process is illustrated in Figure 7.2.

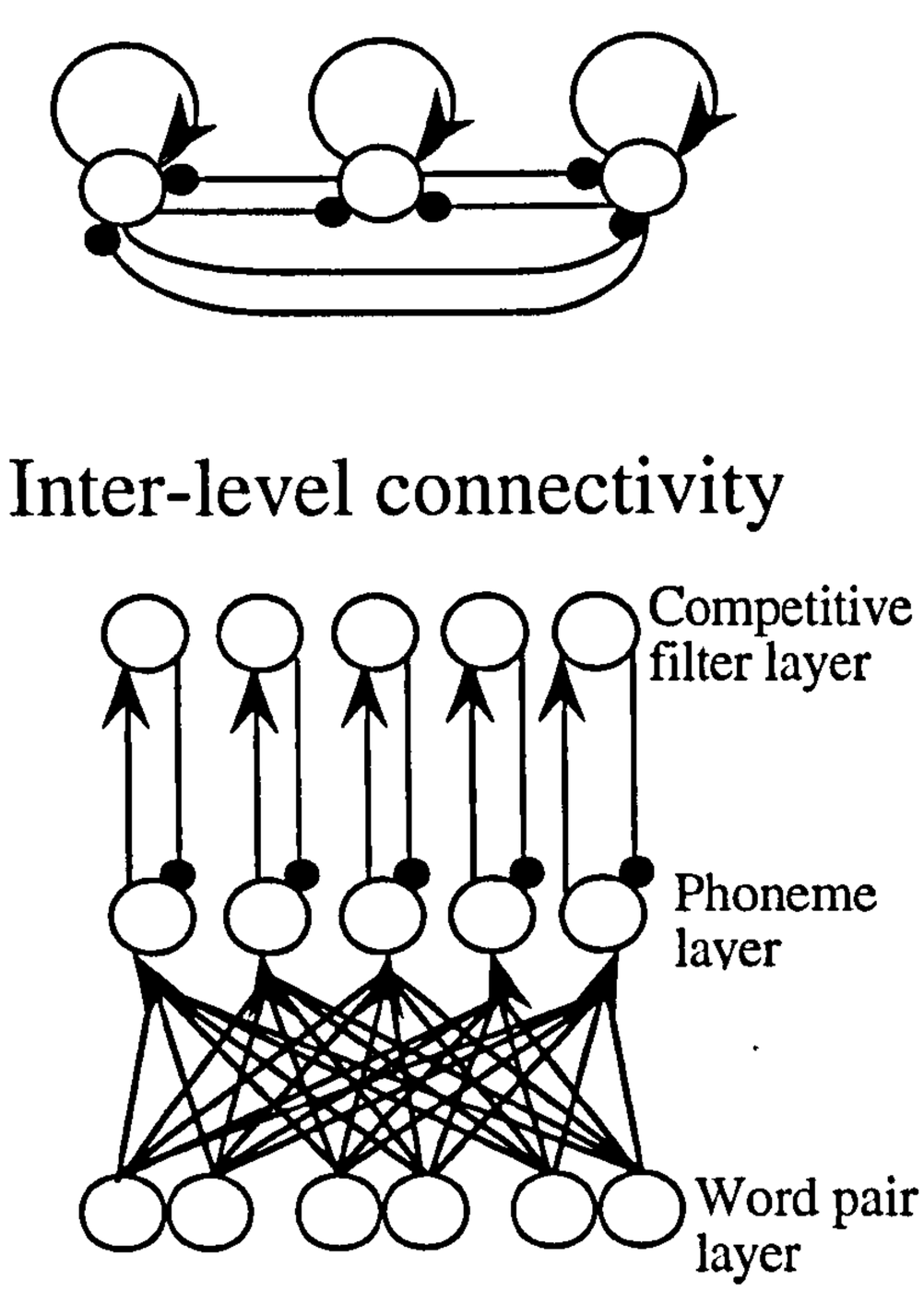

Figure 7.1. Architecture and connectivity in Houghton's (1990) model. Arrows denote excitatory connections, filled circles inhibitory ones.

The simultaneous activation of phonemes formed during recall of the sequence is resolved into the correct serial order in the competitive filter layer. Units compete for selection in this layer through inhibitory connections. This is achieved essentially by selecting and then suppressing the most active unit in turn until the sequence is complete.

In Figure 7.2 the elements in the sequence PROP are systematically associated with updated states of the control signal, indicated by the shaded box. Thus at the start of the sequence P is recalled because it is associated to the first state of the control signal. After it has been output it is suppressed. As the control signal

changes state, so R and O are selected and suppressed in the same way. At the end of the sequence, however, P is reactivated because it is also associated with the end of the control signal. Thus the phoneme layer, where items become simultaneously active before they are output, acts as a sort of short term memory, and the control signal maintains and updates the activation gradient throughout the sequence as it changes state.

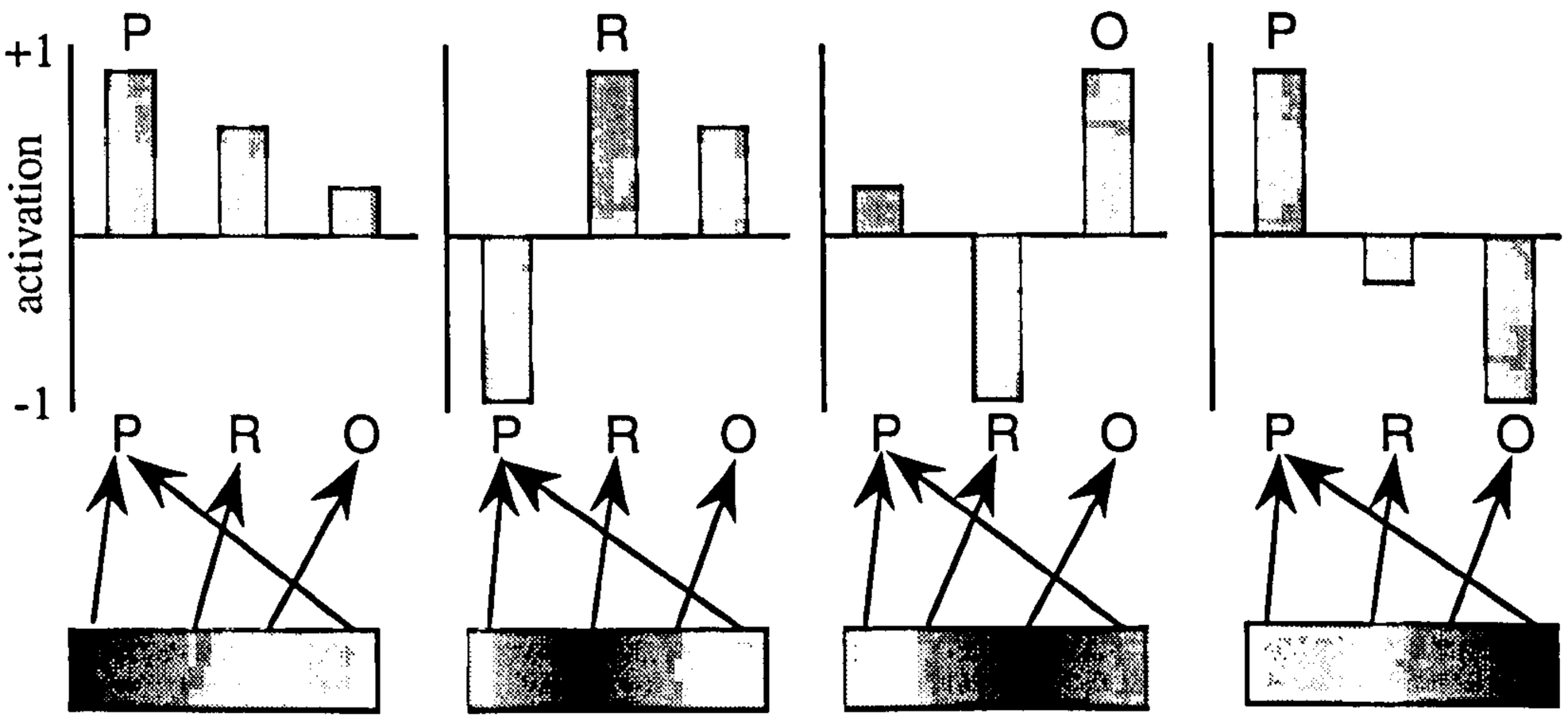

Figure 7.2. Representation of the sequence PROP in Houghton's (1990) CQ model. The shaded boxes represent the dynamic nature of the control signal as it changes state with time, the darkest area being the most active, or current state. Arrows indicate connections to phonemes, and the graphs above show their activation as the context signal changes state.

The notion of a temporally correlated control signal is appealing because the pattern of human errors display a temporal distance effect (MacKay, 1970), although there seems to be little psychological motivation for the control signal in the model. The model is also limited by the low dimensionality (i.e. two) of the control signal. Thus as the length of the sequence increases, the discrimination between adjacent states of the control signal decreases. If some psychological motivation could be found for a control signal that operated in a similar way to that in Houghton's model such that it also provided a more powerful signal (for example by increasing its dimensionality) then perhaps it would be a more viable model. However, reconciling

the serial order of simultaneously represented phonemes suggests CQ models provide a more plausible paradigm for investigating movement errors in speech. I now introduce a model in which the control signal is psychologically motivated by the role of oscillatory systems in maintaining physiological rhythms (e.g. Glass & Mackey, 1988).

7.2. OSCillator based Associative Recall (OSCAR)

An implementation of a model based on the CQ architecture to assess the CQ paradigm seems appropriate. In this section I introduce an improved model based on the CQ paradigm with psychological motivation for a control signal of higher dimensionality, consisting of simple oscillatory devices. The *context signal* performs in a similar manner to Houghton's (1990) control signal, dynamically changing state over time such that it is also temporally correlated. However, because it has greater dimensionality, it is able to discriminate better between adjacent states of sequences. I begin the section with a formal explanation of the construction of the context signal and some of its properties, and continue with a series of simulations. These are designed to explore the properties of the model and demonstrate a computational understanding of how the model resolves the problem of determining serial order from simultaneously represented information.

7.2.1 Oscillator dynamics

For a sequence of items to be learned fluently, the *context* within which each item occurs plays an important role. Consider for example the way in which phonemes are articulated depending on the other phonemes that surround them. The context of phoneme sequences is temporally constrained because nearby phonemes to the currently articulated one influence its precise articulation more so than distant

phonemes. A context signal should therefore exhibit this property, that is its internal similarity should be biased towards nearby states in time.

Recently, research into time perception and rhythmical mechanisms has provided evidence for physiological oscillators as part of a time keeping mechanism in human behaviour (Billon & Semjen, 1995; Glass & MacKay, 1988; Treisman, Cook, Naish & MacCrone, 1994; Treisman, Faulkner & Naish, 1992). An oscillator is a device that moves to and fro within a fixed range in a systematic way (e.g. rotating around an axis), with a periodic frequency. An obvious device to use as part of the temporally correlated context signal seems to be one which is oscillatory based. This can be achieved by constructing a context signal from a vector of temporally changing oscillators. For the purposes of a temporally changing context signal, an oscillator can be defined as a simple processing unit whose activation varies as a function of the angle through which it moves or rotates around an axis. An example of an object that displays oscillatory behaviour is a pendulum. Figure 7.3 shows such an example of a simple oscillatory device.

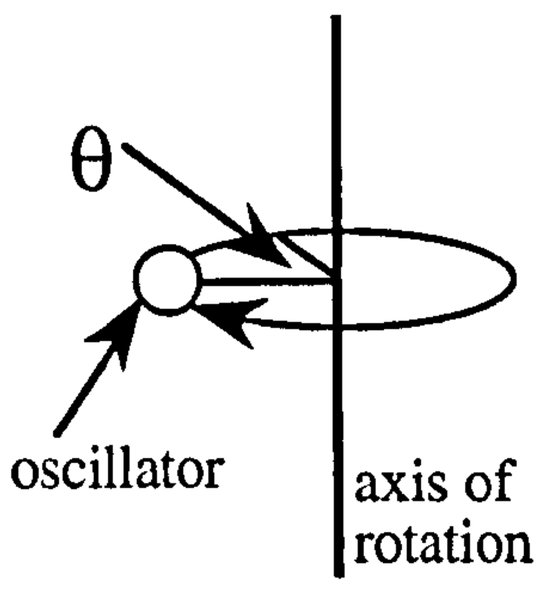

Figure 7.3. An example of a simple oscillatory device that spins around an axis. θ is a measure of the angle through which it rotates. The rate of change of θ determines the speed at which the oscillator rotates around the axis and hence its periodic frequency.

The oscillator is the basic device upon which the context signal is composed.

7.2.2 The Context Signal

The context signal is represented by a multi-dimensional vector, each element consisting of the product of a number of oscillators. The oscillators that constitute the context signal form an arbitrary set, drawn from a larger domain. They oscillate, or rotate at a frequency which can either be different for each oscillator or the same for all. The oscillators might also be out of phase with each other, that is they may be at different angles of rotation. The dynamics of the oscillators are analogous to runners running around a track in lanes. Using the analogy, the oscillators are in phase when the runners all start from the same mark, and the oscillators rotate at the same frequency when the runners all travel at the same speed. Each element of the vector is made up from a fixed number of oscillators, and each oscillator may contribute to one or many elements of the context vector. As the oscillators rotate, each element of the vector is continually updated, and the context as a whole constantly changes state as well. Over a given time period, a dynamic representation of context is generated. Given a large number of oscillators, such a configuration can represent many different contexts.

In order to make use of this dynamic signal, discrete states of the continually changing vector are periodically sampled. The sampling rate of the vector into discrete states is governed by an arbitrary unit of time, henceforth referred to as a *time-step*. Note that sampled states of the vector are temporally correlated, such that they exhibit a systematic similarity relationship.

7.2.2.1 Temporal Correlation

The specific similarity relationship of the context signal depends on the behaviour of the oscillators of which it is made up. If the set of oscillators that contribute to the vector oscillate at the same frequency, but out of phase with one another, then a *clean* context signal is obtained. It is referred to as a clean signal

because the similarity function of the signal is always qualitatively the same for a context signal of fixed dimensions. Each sampled state of the clean signal will have the property that states closer in time will be more similar that states far apart in time, within a given range. This property can be illustrated graphically by plotting a measure of the similarity between states of the clean signal as a function of their separation in time-steps. This is shown in Figure 7.4. Similarity is defined by calculating the dot product between all states and ranges between one (identical states) and zero (lowest measure of similarity).

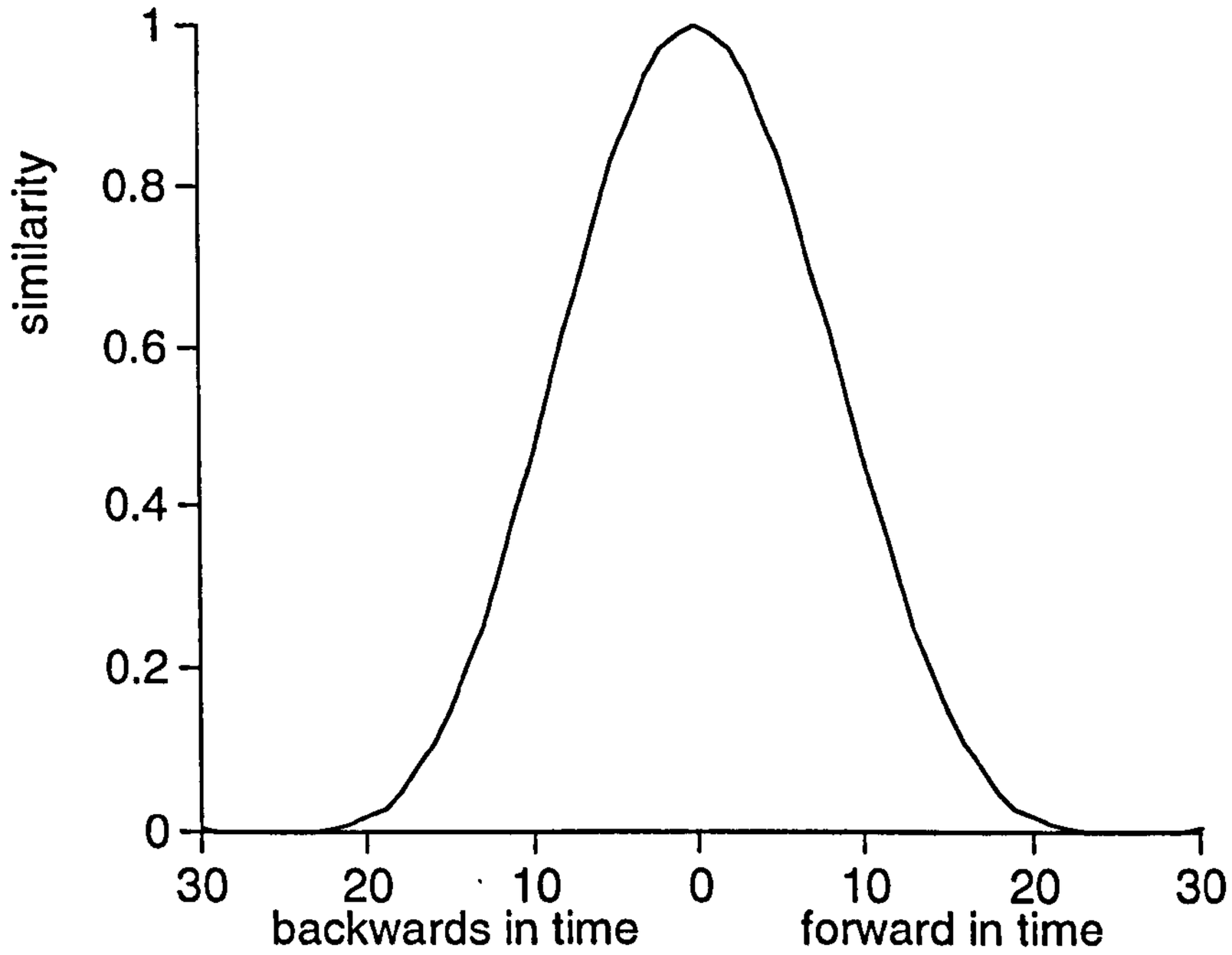

Figure 7.4. Graph depicting the similarity of each sampled state of a clean context vector with every other sampled state. Similarity is defined by calculating the dot product between states. Similarity is plotted as ordinate, against time as abscissa. The closer the sampled states are together in time, the more similar they are.

The sharpness of the peak of the curve in Figure 7.4 illustrates the similarity of temporally separated states. States of the vector with a zero separation have the greatest similarity with each other (i.e. a similarity of one) because they are identical. States that have a separation of say ten, have a lesser similarity with each other, while

even more distant states (say greater than ten) cease to bear any similarity to each other (i.e. similarity of zero) because they are so far apart in time.

7.2.2.2 Mathematical Formulation

The similarity relationship of the signal depicted in Figure 7.4 was calculated from a 16-dimensional clean context signal. A 16-element vector was initially chosen because it was thought that this would be enough to yield a powerful enough signal for the present purpose, while still being sufficiently small to be computationally feasible. The 16-element clean context signal was generated by systematically combining a number of oscillators to make up each element of the vector, such that the vector is normal (Brown, Preece & Hulme, 1996). Specifically, each element was calculated as a combination of four different oscillators, as in (7.1).

$$
\begin{aligned}
&\text{context\_element}(1) = \cos(\theta_1) * \cos(\theta_2) * \cos(\theta_3) * \cos(\theta_4);\\
&\text{context\_element}(2) = \cos(\theta_1) * \cos(\theta_2) * \cos(\theta_3) * \sin(\theta_4);\\
&\text{context\_element}(3) = \cos(\theta_1) * \cos(\theta_2) * \sin(\theta_3) * \cos(\theta_5);\\
&\text{context\_element}(4) = \cos(\theta_1) * \cos(\theta_2) * \sin(\theta_3) * \sin(\theta_5);\\
&\text{context\_element}(5) = \cos(\theta_1) * \sin(\theta_2) * \cos(\theta_6) * \cos(\theta_7);\\
&\text{context\_element}(6) = \cos(\theta_1) * \sin(\theta_2) * \cos(\theta_6) * \sin(\theta_7);\\
&\text{context\_element}(7) = \cos(\theta_1) * \sin(\theta_2) * \sin(\theta_6) * \cos(\theta_8);\\
&\text{context\_element}(8) = \cos(\theta_1) * \sin(\theta_2) * \sin(\theta_6) * \sin(\theta_8);\\
&\text{context\_element}(9) = \sin(\theta_1) * \cos(\theta_9) * \cos(\theta_{10}) * \cos(\theta_{11});\\
&\text{context\_element}(10) = \sin(\theta_1) * \cos(\theta_9) * \cos(\theta_{10}) * \sin(\theta_{11});\\
&\text{context\_element}(11) = \sin(\theta_1) * \cos(\theta_9) * \sin(\theta_{10}) * \cos(\theta_{12});\\
&\text{context\_element}(12) = \sin(\theta_1) * \cos(\theta_9) * \sin(\theta_{10}) * \sin(\theta_{12});\\
&\text{context\_element}(13) = \sin(\theta_1) * \sin(\theta_9) * \cos(\theta_{13}) * \cos(\theta_{14});\\
&\text{context\_element}(14) = \sin(\theta_1) * \sin(\theta_9) * \cos(\theta_{13}) * \sin(\theta_{14});\\
&\text{context\_element}(15) = \sin(\theta_1) * \sin(\theta_9) * \sin(\theta_{13}) * \cos(\theta_{15});\\
&\text{context\_element}(16) = \sin(\theta_1) * \sin(\theta_9) * \sin(\theta_{13}) * \sin(\theta_{15});
\end{aligned}
\qquad (7.1)
$$

Each oscillator in (7.1) is denoted by θ, the subscript referring to the particular oscillator being described. The advantage of constructing a context signal in this way is that each state of the signal is represented by a normalised vector; that is the dot product of the vector with itself is one. This is useful when comparing states of the context to each other because states with maximum similarity will have a dot product of one, and less similar states will have dot products less than one.

The starting position of each oscillator was set at random, thus ensuring that they were out of phase with each other. All oscillators were set to rotate at the same frequency, of 0.06 radians per time-step. For the first time-step, each element of the vector was calculated as above in (7.1). Depending on the frequency of the oscillators, the value of the context vector on subsequent time-steps (again calculated according to (7.1)) was different (although still related) to previous states because the oscillators had changed position.

There is no reason (aside from computational considerations) why the context signal should be constrained to 16-element vector, as in (7.1), providing that it yields the same general similarity relationship as just presented.

7.2.2.3 Vector Dimensionality and the Similarity Relationship

To test whether the general similarity relationship holds for context signals of a higher dimensionality, a vector was constructed that consisted of a greater number of elements and a greater number of oscillators. A 32-element vector was the next size context signal that could be constructed using the same method as in (7.1) such that the signal was normal. Using the same method for calculating successive states of the context signal, the similarity relationship of a 32-element context signal was calculated. Each element in the vector consisted of the combination of five different oscillators, as in (7.2).

$$
\begin{aligned}
\text{context\_element}(1) &= \cos(\theta_1) * \cos(\theta_2) * \cos(\theta_4) * \cos(\theta_8) * \cos(\theta_{16});\\
\text{context\_element}(2) &= \cos(\theta_1) * \cos(\theta_2) * \cos(\theta_4) * \cos(\theta_8) * \sin(\theta_{16});\\
\text{context\_element}(3) &= \cos(\theta_1) * \cos(\theta_2) * \cos(\theta_4) * \sin(\theta_8) * \cos(\theta_{17});\\
\text{context\_element}(4) &= \cos(\theta_1) * \cos(\theta_2) * \cos(\theta_4) * \sin(\theta_8) * \sin(\theta_{17});\\
\text{context\_element}(5) &= \cos(\theta_1) * \cos(\theta_2) * \sin(\theta_4) * \cos(\theta_9) * \cos(\theta_{18});\\
\text{context\_element}(6) &= \cos(\theta_1) * \cos(\theta_2) * \sin(\theta_4) * \cos(\theta_9) * \sin(\theta_{18});\\
\text{context\_element}(7) &= \cos(\theta_1) * \cos(\theta_2) * \sin(\theta_4) * \sin(\theta_9) * \cos(\theta_{19});\\
\text{context\_element}(8) &= \cos(\theta_1) * \cos(\theta_2) * \sin(\theta_4) * \sin(\theta_9) * \sin(\theta_{19});\\
\text{context\_element}(9) &= \cos(\theta_1) * \sin(\theta_2) * \cos(\theta_5) * \cos(\theta_{10}) * \cos(\theta_{20});\\
\text{context\_element}(10) &= \cos(\theta_1) * \sin(\theta_2) * \cos(\theta_5) * \cos(\theta_{10}) * \sin(\theta_{20});\\
\text{context\_element}(11) &= \cos(\theta_1) * \sin(\theta_2) * \cos(\theta_5) * \sin(\theta_{10}) * \cos(\theta_{21});\\
\text{context\_element}(12) &= \cos(\theta_1) * \sin(\theta_2) * \cos(\theta_5) * \sin(\theta_{10}) * \sin(\theta_{21});\\
\text{context\_element}(13) &= \cos(\theta_1) * \sin(\theta_2) * \sin(\theta_5) * \cos(\theta_{11}) * \cos(\theta_{22});\\
\text{context\_element}(15) &= \cos(\theta_1) * \sin(\theta_2) * \sin(\theta_5) * \sin(\theta_{11}) * \cos(\theta_{23});\\
\text{context\_element}(16) &= \cos(\theta_1) * \sin(\theta_2) * \sin(\theta_5) * \sin(\theta_{11}) * \sin(\theta_{23});\\
\text{context\_element}(17) &= \sin(\theta_1) * \cos(\theta_3) * \cos(\theta_6) * \cos(\theta_{12}) * \cos(\theta_{24});\\
\text{context\_element}(18) &= \sin(\theta_1) * \cos(\theta_3) * \cos(\theta_6) * \cos(\theta_{12}) * \sin(\theta_{24});\\
\text{context\_element}(19) &= \sin(\theta_1) * \cos(\theta_3) * \cos(\theta_6) * \sin(\theta_{12}) * \cos(\theta_{25});\\
\text{context\_element}(20) &= \sin(\theta_1) * \cos(\theta_3) * \cos(\theta_6) * \sin(\theta_{12}) * \sin(\theta_{25});\\
\text{context\_element}(21) &= \sin(\theta_1) * \cos(\theta_3) * \sin(\theta_6) * \cos(\theta_{13}) * \cos(\theta_{26});\\
\text{context\_element}(22) &= \sin(\theta_1) * \cos(\theta_3) * \sin(\theta_6) * \cos(\theta_{13}) * \sin(\theta_{26});\\
\text{context\_element}(23) &= \sin(\theta_1) * \cos(\theta_3) * \sin(\theta_6) * \sin(\theta_{13}) * \cos(\theta_{27});\\
\text{context\_element}(24) &= \sin(\theta_1) * \cos(\theta_3) * \sin(\theta_6) * \sin(\theta_{13}) * \sin(\theta_{27});\\
\text{context\_element}(25) &= \sin(\theta_1) * \sin(\theta_3) * \cos(\theta_7) * \cos(\theta_{14}) * \cos(\theta_{28});\\
\text{context\_element}(26) &= \sin(\theta_1) * \sin(\theta_3) * \cos(\theta_7) * \cos(\theta_{14}) * \sin(\theta_{28});\\
\text{context\_element}(27) &= \sin(\theta_1) * \sin(\theta_3) * \cos(\theta_7) * \sin(\theta_{14}) * \cos(\theta_{29});\\
\text{context\_element}(28) &= \sin(\theta_1) * \sin(\theta_3) * \cos(\theta_7) * \sin(\theta_{14}) * \sin(\theta_{29});\\
\text{context\_element}(29) &= \sin(\theta_1) * \sin(\theta_3) * \sin(\theta_7) * \cos(\theta_{15}) * \cos(\theta_{30});\\
\text{context\_element}(30) &= \sin(\theta_1) * \sin(\theta_3) * \sin(\theta_7) * \cos(\theta_{15}) * \sin(\theta_{30});\\
\text{context\_element}(31) &= \sin(\theta_1) * \sin(\theta_3) * \sin(\theta_7) * \sin(\theta_{15}) * \cos(\theta_{31});\\
\text{context\_element}(32) &= \sin(\theta_1) * \sin(\theta_3) * \sin(\theta_7) * \sin(\theta_{15}) * \sin(\theta_{31});
\end{aligned}
\tag{7.2}
$$

As can be seen from Figure 7.5, the general principle still holds, but the range of the similarity measure now extends from plus one down to minus one. Also, the qualitative shape of the similarity function is different from the similarity function of the 16-element context signal. States further apart in time (or separation) are still less similar than nearby states, but the fact that some states have a similarity measure of less than one indicates that although states of the context signal are equal in magnitude (because they are normal) they are pointing in opposite directions.

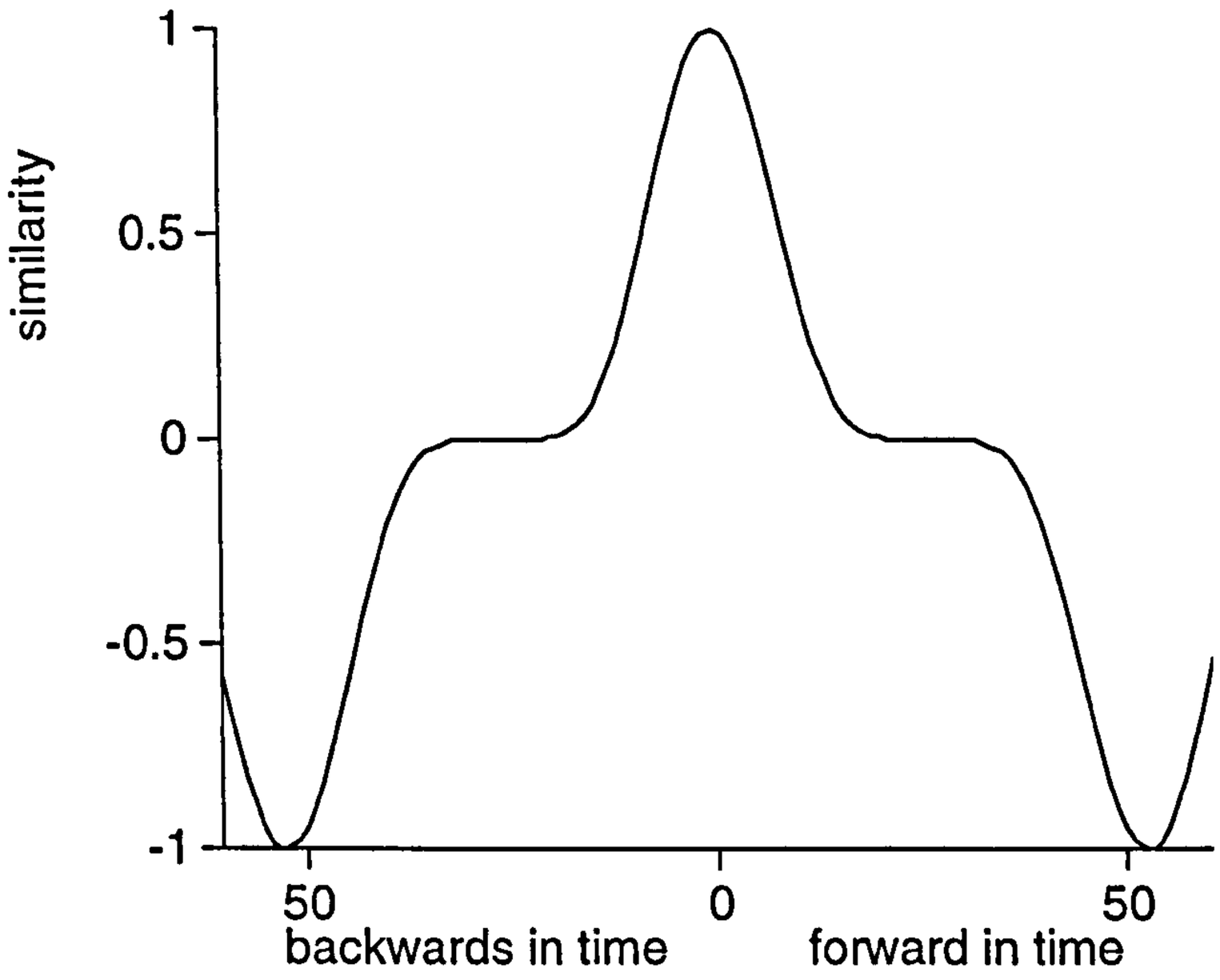

Figure 7.5. Graph depicting the similarity of each sampled state of the 32-element context signal with every other sampled state. Similarity is plotted as ordinate, against time as abscissa. The closer the sampled states are together in time, the more similar they are.

This can be explained using basic mathematical principles. For a 16-element vector, each element is calculated as the product of a combination of the cosine or sine of four oscillators. As each oscillator rotates through the first half of its period (i.e. π radians), its sine value is positive, and then turns negative in the second half. Similarly, its cosine is positive in the first and last quarter of rotation and negative in the semi-interquartile range. Thus each element may take a positive or negative value, depending on the particular value of each oscillator that contributes to it. Given also that each oscillator is spinning with the same frequency, after a certain amount of time all the oscillators will have rotated through exactly π radians, or one half rotation and therefore its sine or cosine value is guaranteed to be equal in magnitude yet opposite in sign. However, because each element of the context is calculated as the product of a combination of the cosine or sine of *four* oscillators, the sign of each *element* remains the same after π radians of rotation. For example,

suppose the first element is calculated as in (7.1), where each θ has the following values:

context_element(1) = cos(π/8) * cos(π/4) * cos(π) * cos(4π /3);
= 0.9238 * 0.7071 * -1 * -0.5
= 0.3266

Now suppose that a certain amount of time has elapsed such that each oscillator has rotated through π radians. The calculation of the same (first) element of the context is now as follows:

context_element(1) = cos(9π/8) * cos(5π/4) * cos(2π) * cos(7π /3);
= -0.9238 * -0.7071 * 1 * 0.5
= 0.3266

The point to be noted here is that although each constituent part of the element changes sign, the sign of the element itself remains *unchanged* because the multiplication of an even number of values of the same sign is a result which has equal sign. Thus for elements comprising four elements, the number of positive and negative components will remain either odd or even after π radians of oscillation.

The difference between the similarity plots for 16-element and 32-element contexts can thus be attributed to the number of constituent parts that make up each element. For a 32-element context, there are five oscillators contributing to each element. Thus after π radians of oscillation, the cosine (or sine) of each oscillator will be opposite in sign and due to the odd number of constituent parts, the sign of the resulting element will now also be different. This effect holds for vectors of dimensions for which their constituent elements consist of an odd number of oscillators, for example for an eight element vector whose elements consist of a combination of three oscillators. This sort of characteristic of the context vector has

implications for its use as a vector with which to associate sequential information. Specifically, if states of the context signal whose similarity is -1 are associated with similar items, then learning the second item can cause the first item to be lost. I shall expand on this point in the next section.

7.2.2.4 Vector Dimensionality and Complex Sequences

Repetition of items within a sequence causes severe (if not catastrophic) problems for context vectors whose similarity relationship looks like Figure 7.5. As was mentioned earlier, the magnitude of these vectors after each oscillator has rotated through π radians remains the same, but their direction is completely opposite, indicated by a similarity measure of minus one. That is, each element of the vector has the same magnitude but opposite sign. Imagine associating the context signal with a sequence of items in which item A appears twice. If, when the second occurrence of item A is associated to the context signal, each oscillator has rotated through π radians, then the item A will effectively become 'unlearnt' due to the dynamics of the context. For example, the actual values of a four-element context signal might initially have the following values, when the first occurrence of item 'A' is associated with it:

element(1) = 0.23
element(2) = -0.16
element(3) = 0.69
element(4) = -0.52

After item A has been associated with that state of the context signal, the weights connecting that state and item A will be incremented with the initial values of the context shown above. However, after π radians, (when the second occurrence of item 'A' is associated to the context signal), the elements of the current state of the context will now look like this:

element(1) = -0.23
element(2) = 0.16
element(3) = -0.69
element(4) = 0.52

Associating the second occurrence of item A with the context signal after π radians results in incrementing the same set of weights with the above values of the new state of the context signal. However, these values are opposite in sign and equal in magnitude. Incrementing the same set of weights with values equal in magnitude and opposite in sign to their current value effectively reduces the weights to zero and causes 'unlearning' of the first occurrence of item A. Hence caution must be taken when using a context signal whose similarity measure drops significantly below zero. It would seem wise, therefore, to use a context signal whose elements are constructed from the product of an even number of oscillators, to avoid these problems.

7.2.2.5 Effects of Oscillator Frequency on the Clear Context Signal

The 16-element configuration produces a periodic context signal, that is a signal which recurs at regular intervals. When the oscillators rotate at a frequency of 0.06 radians per time-step (radians were used as opposed to degrees for ease of computation), the 16-dimensional clean context signal has a period of approximately 32 time-steps (i.e. before the signal starts to repeat previous states, or recur). If the frequency of the oscillators is set at a faster rate, say 0.12 radians per time-step (double the original rate of 0.06 radians per time-step), then the state of the signal on each successive time-step will have changed more. Hence successive states of a signal generated from high frequency oscillators will be less similar than successive states of a context signal generated from low frequency oscillators. A context signal generated from high frequency oscillators will therefore recur after fewer time-steps than a low frequency signal. The graph in Figure 7.6 shows the similarity relationship of the same 16-dimensional clean signal, when constructed from high

frequency oscillators. The shape of the similarity function is qualitatively the same as in Figure 7.4, when low frequency oscillators were used, but the period of the signal is shortened.

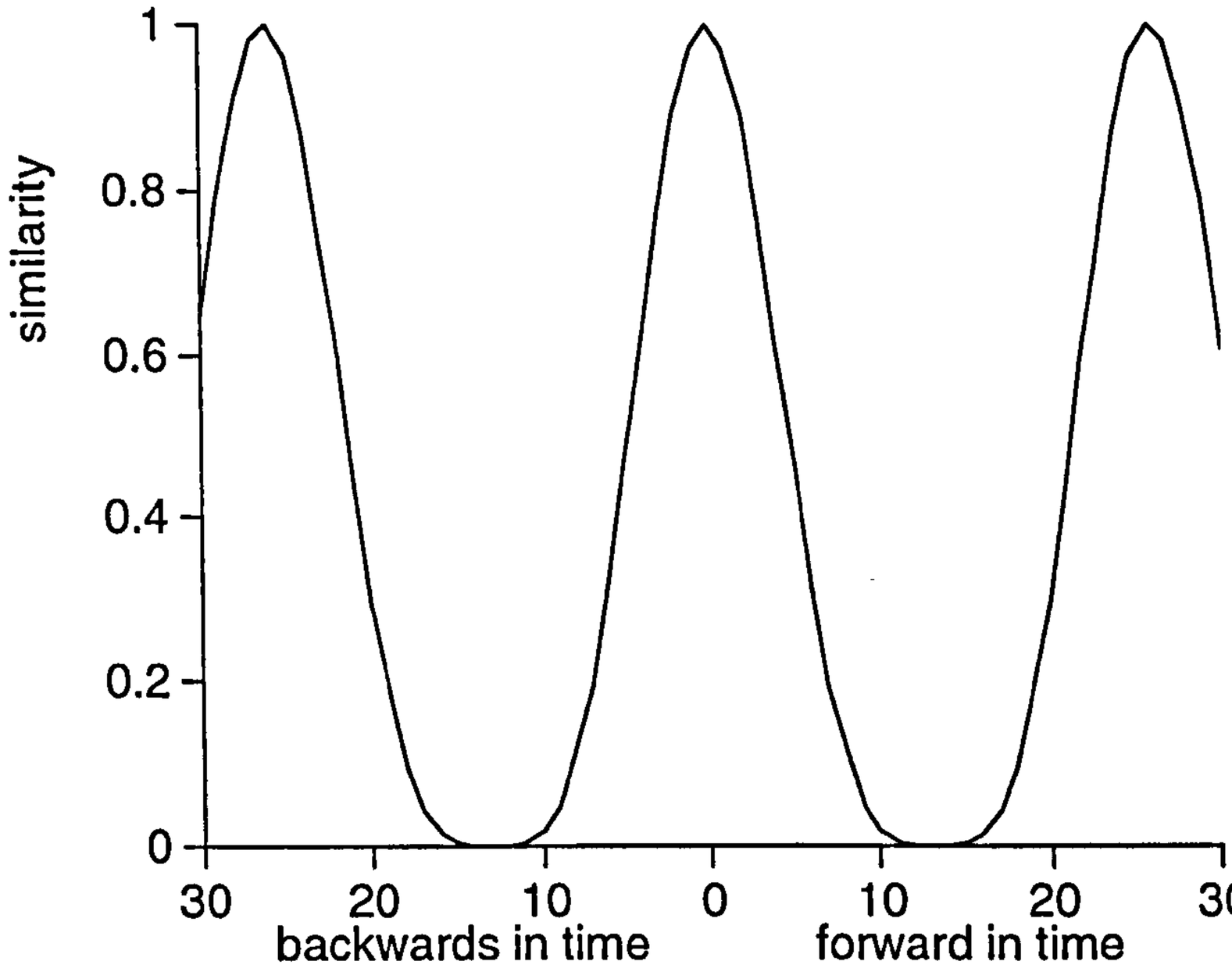

Figure 7.6. Graph depicting the similarity of each sampled state of the context signal with every other sampled state when the signal contains high frequency oscillators. Similarity is plotted as ordinate, against time as abscissa. Note adjacent states are more similar than adjacent states of the context signal illustrated in Figure 7.4.

The sharpness of the peaks in the graph in Figure 7.6 show that successive states of the context signal are less similar than successive states of the context signal generated from low frequency oscillators (see Figure 7.4). Figure 7.6 is analogous to the similarity function that would be obtained by plotting the similarity of every second time-step (or degree of separation) in Figure 7.4. Thus the graph in Figure 7.4 can be thought of as the general similarity relationship generated when the clean context signal is updated every time-step. Updating the state of the context signal on every second, or third time-step is analogous to increasing the frequency of the oscillators of which it is composed. The simulations that follow use the context

signal generated with oscillators rotating at a frequency of 0.06 radians per time-step. For ease of reference, I shall use the term *step-size* to refer to the number of time-steps that elapse before the next state of the context signal is calculated. For example, the graph in Figure 7.4 shows the similarity function of a context signal with a step-size of one, and the similarity function obtained for a context signal with a step-size of two is shown in Figure 7.6.

7.3. Computer Simulations of OSCAR

The basic properties of the OSCAR model can be explored by way of computer simulations. Using the 16-dimensional clean context signal, sequences can be learned by associating each item in the sequence with successive states of the context signal as it changes over time. Throughout the rest of this chapter, all the computer simulations are based on a simple Hebbian network paradigm using the 16-dimensional clean context signal. The task to be demonstrated is to learn a sequence of phonemes and then to recall them in the correct order when provided with the appropriate context signal as a cue. Two basic models are simulated. The first model assumes the phonemes in the sequence are represented by individual, independent units. The second model explores the consequences of assuming a distributed coding scheme for phonemes. Both models are tested on their ability to represent phoneme sequences with no repeated information (*simple sequences*), and sequences that contain repeated phonemes (*complex sequences*). The step-size of the context signal was varied in both models.

7.3.1 Model 7.1

Model 7.1 was designed to illustrate the basic similarity principle of OSCAR. This principle states that nearby phonemes in time are more similar and hence nearby phonemes will be more co-activated than distant ones.

The first task for Model 7.1 was to learn to recall a simple sequence of phonemes, such as '/k r I s p/'.

7.3.1.1 Architecture

The first model assumed a local coding scheme to represent the phonemes. Specifically, each possible phoneme is represented by an individual output node that is solely dedicated to the activation of just that phoneme. Figure 7.7 shows the architecture of Model 7.1 using a local coding scheme to represent the output phonemes. The bottom layer of units in this case represents the 16-element context signal and the top layer represents all possible phonemes. There is exactly one unit for each phoneme. Lines connecting the two layers represent weighted connections from the context signal to the phoneme nodes and every element of the context signal is connected to every phoneme node.

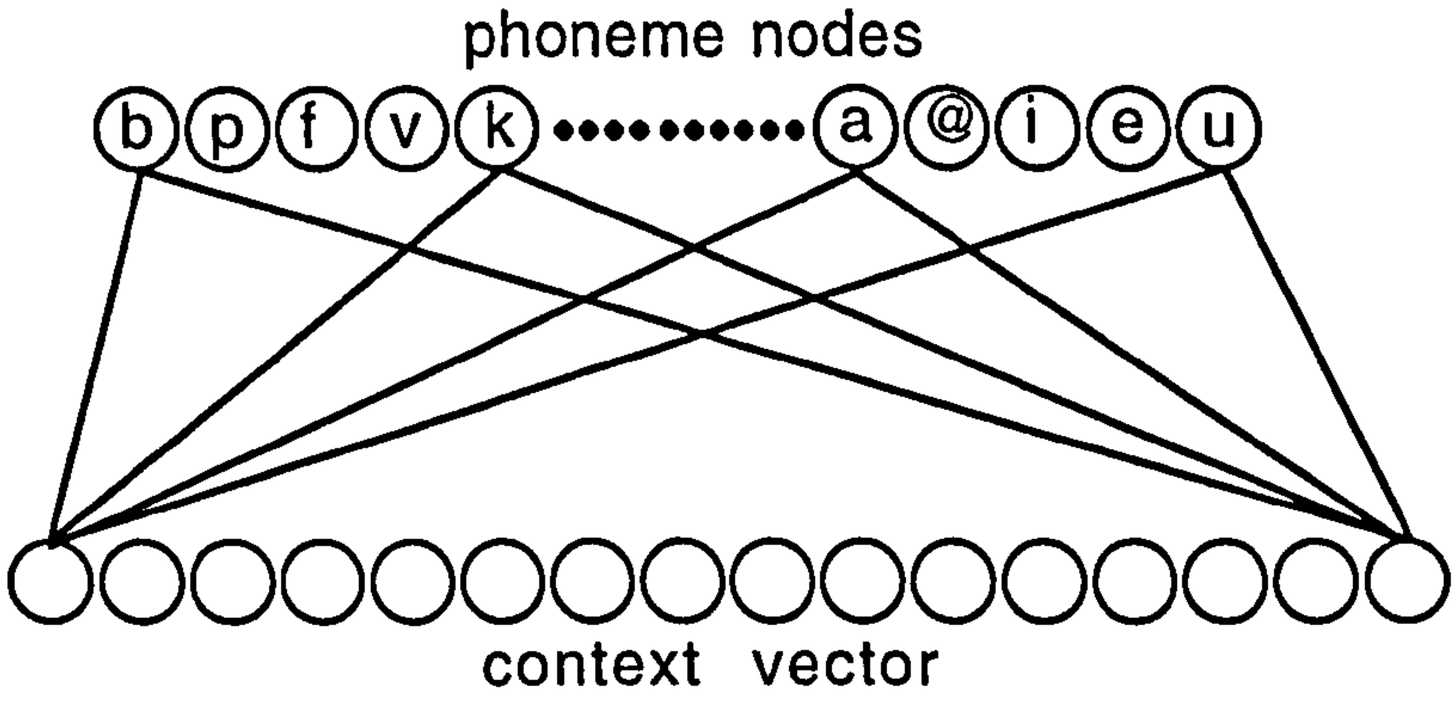

Figure 7.7. The architecture of the OSCAR model using a 16-element context signal. Circles in the bottom row represent each element of the vector in the context signal and the top row of units represent each phoneme. Not all nodes in the phoneme layer have been included in the diagram. Lines connecting the two layers represent weighted connections, but for the sake of clarity only a few arbitrary connections are shown.

7.3.1.2 Method

Each phoneme was represented by an individual node and was connected by a link, or weight, to every element in the context signal, as illustrated in Figure 7.7. Initially every weight was set to zero. The context signal was initialised by selecting a set of oscillators such that each one was assigned a different starting position at random. This vector represented the context associated with the particular sequence of phonemes to be learned. The step-size of the context vector during learning was varied to illustrate the similarity principle. Hebbian learning was then implemented as follows.

7.3.1.3 Learning Algorithm

For each phoneme in the to-be-learned sequence, the appropriate node was fully activated to one whilst the other phonemes' activation remained at zero, and the weights connecting the phoneme units to the context were updated according to (7.3) and (7.4),

$$\Delta w_{(cxt)p}(t) = a_{cxt} a_p \alpha \tag{7.3}$$

$$w_{(cxt)p}(t+1) = w_{(cxt)p}(t) + \Delta w_{(cxt)p} \tag{7.4}$$

where $w_{(cxt)p}$ represents the weight connecting a context and phoneme node, a_{cxt} represents the activation of a context node, a_p represents the activation of a phoneme node, and α represents the rate of learning, in this case set to one. Each phoneme in a sequence was associated to a new state of the context signal. The step-size controlled how many time-steps elapsed before the context signal was updated. In this case, the step-size was one. The whole process was repeated until all the phonemes had been learned.

7.3.1.4 Recall

When a sequence of phonemes was to be recalled, the appropriate context signal was reinstated. Activation flowed from the context signal along the weighted links to the phoneme nodes. The activation of the phoneme nodes was simply calculated as the sum of the product of the activation of each context node and the weight of the link connecting it to the phoneme node, as in (7.5).

$$a_p = \sum a_{cxt} w_{(cxt)p} \tag{7.5}$$

As the context signal was updated, a new pattern of activation on the phoneme nodes was calculated accordingly, the most highly activated node represented the phoneme most likely to be output at that point in time.

7.3.1.5 Results of Preliminary Explorations

7.3.1.5.1 Learning a Simple Sequence

Figure 7.8 shows the activation gradient of the appropriate phoneme nodes when Model 7.1 was taught the simple phoneme sequence '/k r I s p/' by associating each phoneme to a successive state of the context, with a step-size of one.

All the phonemes that form part of the sequence were highly activated throughout the time course of production, with their activation level never falling below 0.85. At each step in the production process the network had learned to recall the appropriate phoneme, even though competition from the other phonemes was high.

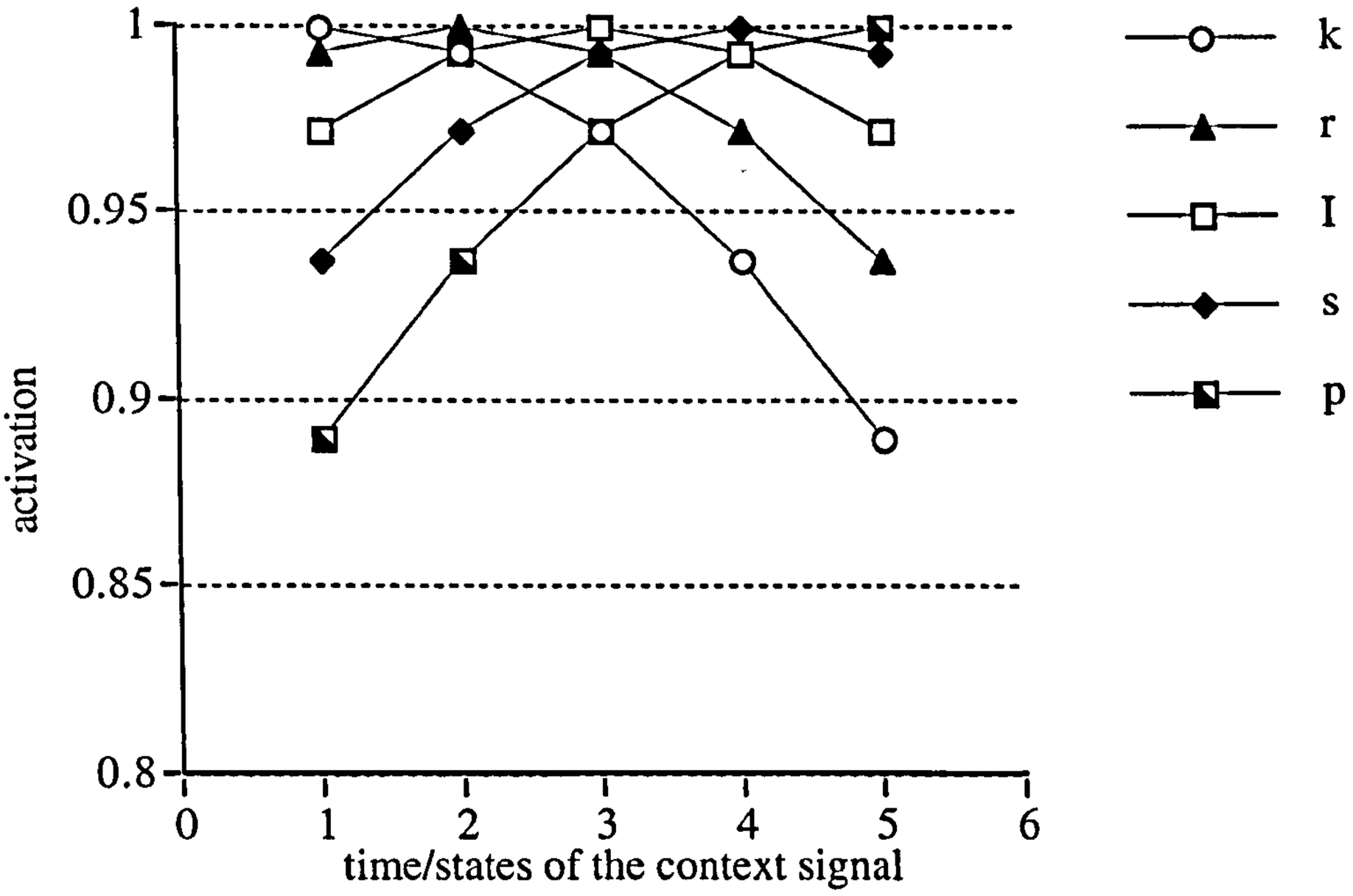

Figure 7.8. Activation gradients of phoneme nodes from Model 7.1 with a step-size of one, over five successive states of the context signal for the sequence '/k r I s p/'.

Figure 7.9 shows a similar activation plot for the same sequence when Model 7.1 was taught the same phoneme sequence, using a context signal with a step-size of three.

Using a larger step-size, each phoneme was effectively learnt further apart in time and therefore its activation level changed more rapidly throughout the production of the sequence. As can be seen, the competing phonemes in Figure 7.9 are less active than the competing phonemes shown in Figure 7.8. This effect is amplified for non-adjacent phonemes throughout the sequence. As the graph in Figure 7.9 shows, the activation of maximally separated phonemes falls to less than 0.4. This is because by using a step-size of three the effect of diminishing activation over time is multiplied by a factor of three.

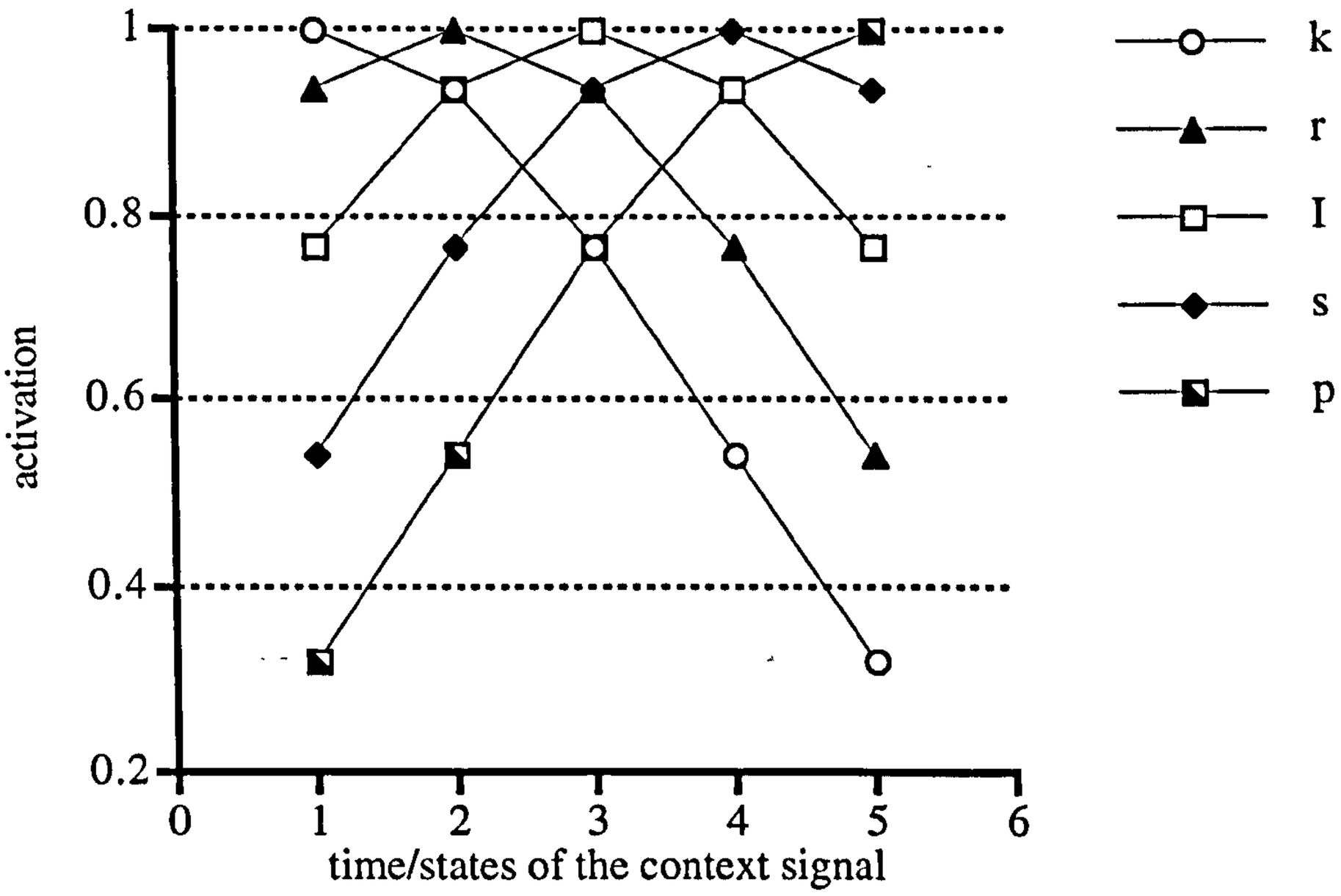

Figure 7.9. Activation gradients of phoneme nodes from Model 7.1 with a step-size of three, over five successive states of the context signal for the sequence '/k r I s p/'.

7.3.1.5.2 Learning a Complex Sequence

Model 7.1 was also trained on a complex sequence; in this case the sequence '/k L I k/'. Figure 7.10 shows the activation of the phoneme sequence '/k L I k/' when the sequence was learned using a context signal with a step-size of three.

As can be seen from the graph, sequences containing repeated phonemes are not well recalled by this model, and the activation of the repeated phoneme remains the most highly activated during inappropriate time intervals. This is illustrated in Figure 7.10, where the activation of phoneme /k/ remains active throughout the time course of production. Although the pattern of the activation gradient for phoneme nodes /L/ and /I/ shows that they would normally be recalled in the correct order and at the appropriate time, their activation levels are swamped by the over active /k/ node.

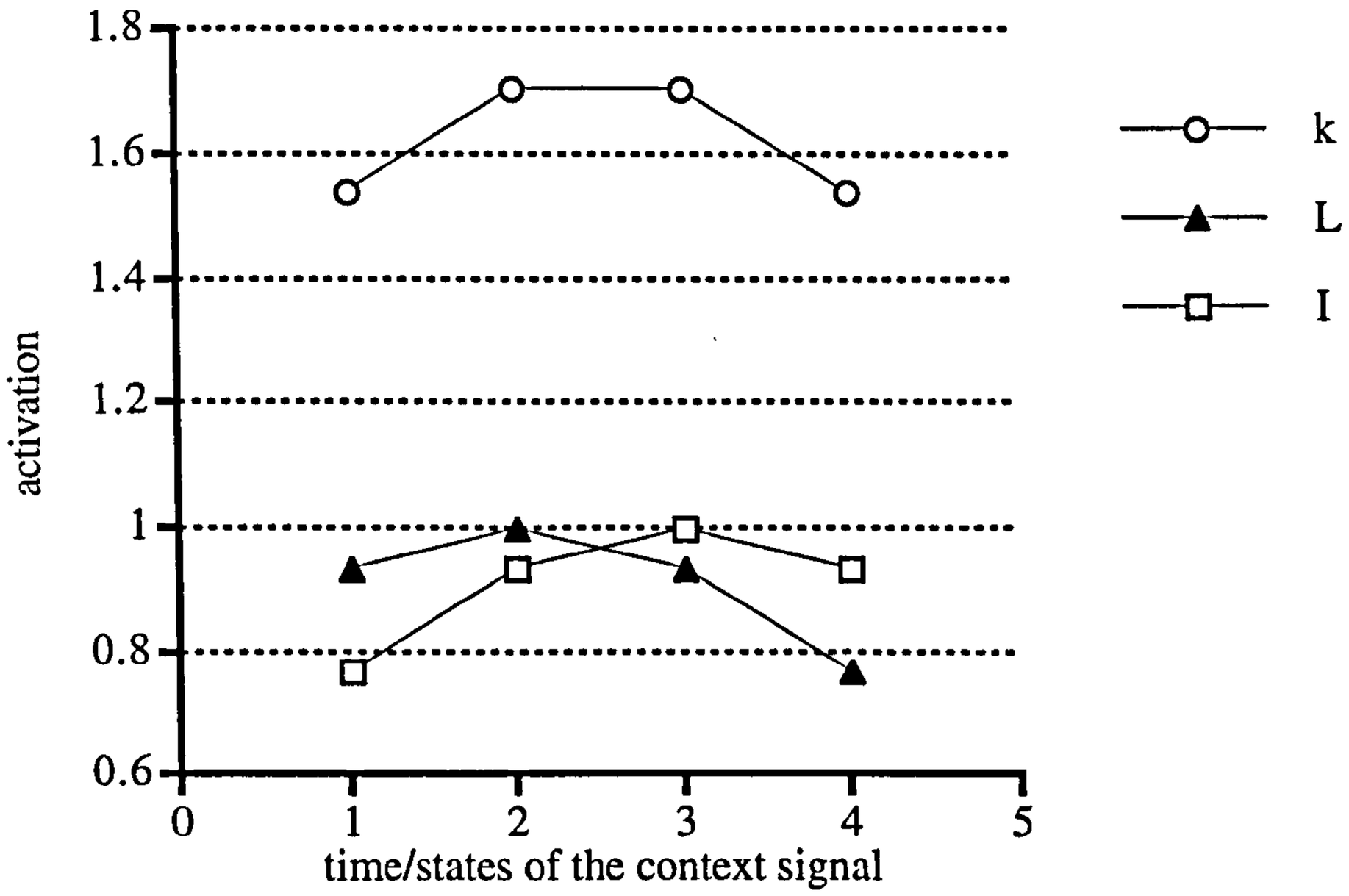

Figure 7.10. Activation gradients of phoneme nodes from Model 7.1 with a step-size of three, over five successive states of the context signal for the sequence '/k L I k/'.

However, different results are obtained when a context signal with a step-size of seven is associated with adjacent phonemes, as shown in Figure 7.11.

Figure 7.11 shows the same pattern of activation gradients for the /L/ and /I/ phoneme nodes as was seen in Figure 7.10, but their activation over time now varies much more, and more importantly the activation of the /k/ node falls sufficiently during mid-sequence for /L/ and /I/ to be correctly recalled. This can be explained by the similarity principle of nearby states. When adjacent phonemes are associated to nearby states of the context signal, as in Figure 7.10, the activation of each phoneme during recall will be more similar to their neighbouring phonemes than when adjacent phonemes are associated to more distant states of the context signal, as in Figure 7.11.

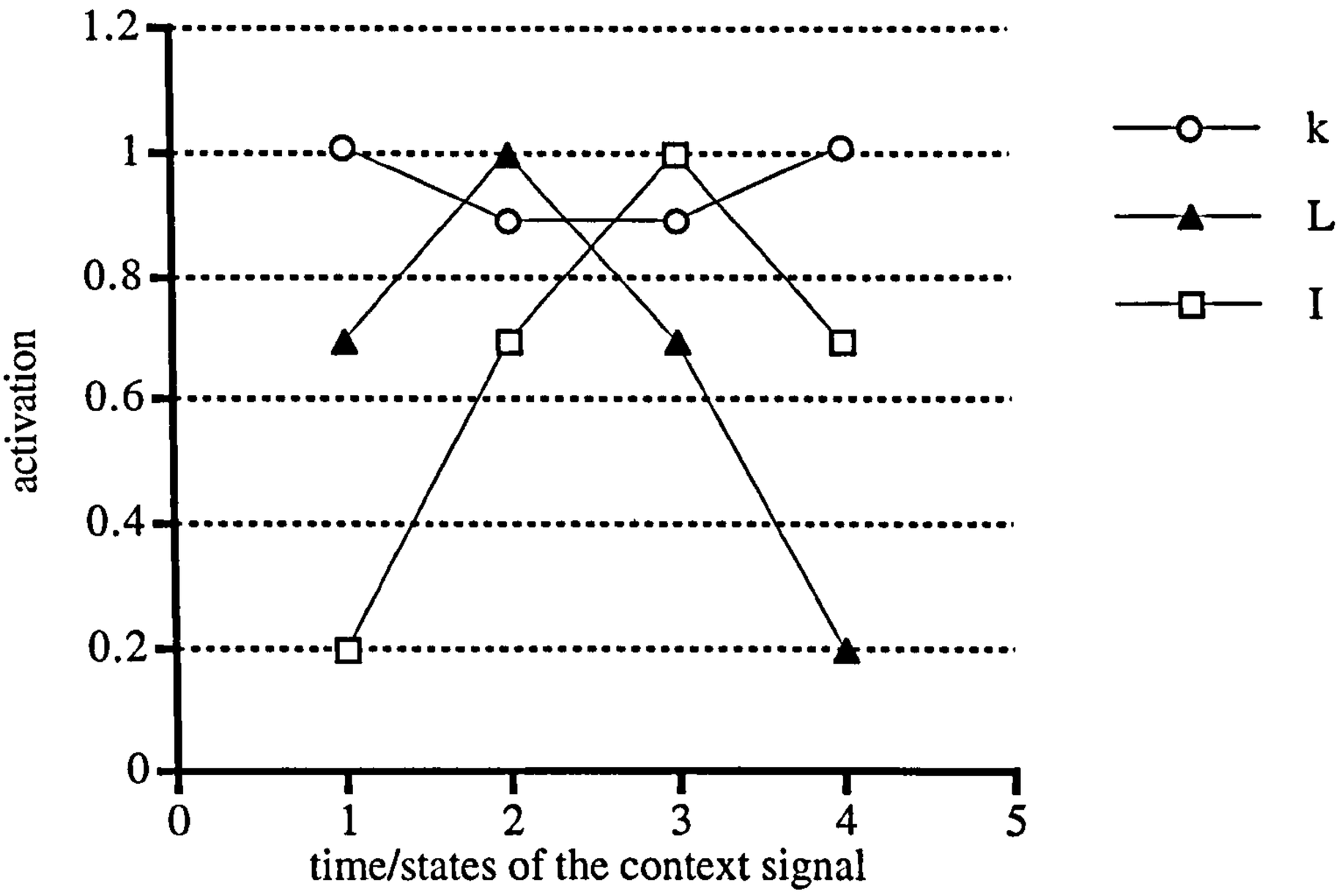

Figure 7.11. Activation gradients of phoneme nodes from Model 7.1 with a step-size of seven, over five successive states of the context signal for the sequence '/k L I k/'.

Thus for the data in Figure 7.10, when the step-size was three, the time course of activation of the /k/ node is explained as follows. On initial reinstatement of the context signal, the /k/ node is most active because the context matches that for the output of /k/, and the /L/ and /I/ node both become partially active because they are temporally close to /k/. For the next state of the context signal, the context matches that for the output of /L/, but /k/ is output instead. This is because /L/ is temporally close to both the first and second production of /k/. Thus /k/ is partially activated by two temporally close states of the context signal. These partial activations are additive and together produce a higher activation than the /L/ node. The same holds true for the state of the context signal and the /I/ node.

However, when a step-size of seven is used during learning, the effective separation between the two /k/ nodes is large enough that the two productions of /k/ appear temporally more distant, and the /L/ and /I/ node are no longer swamped by both productions of the /k/ node and are correctly recalled. Their activation varies

over a greater range during production for the same reason, i.e. because the larger step-size during training has resulted in a greater temporal separation between phonemes.

7.3.1.6 Discussion

The first simulation of Model 7.1 showed how the activation of phonemes varies as a function of their temporal position in the sequence. Thus a phonemes' activation rises before output, and falls after, reaching maximum activation at the time of production. A direct consequence of this is that neighbouring phonemes within the sequence are more active than non-neighbouring or distant phonemes.

By varying the step-size of the context signal during learning, I have demonstrated how complex phoneme sequences can be learned and recalled with a simple model of OSCAR. The results so far illustrate how successive states of a context signal can be associated with adjacent phonemes in a sequence, where the sequence is either a simple string of non-repeated items, or a complex one with repeated phonemes. The smaller the step-size, the more similar the states of the context signal, and hence the more highly co-activated temporally close phonemes are. Larger step-sizes are needed when a sequence contains repeated items to further separate the contexts of the repeated items in time. This prevents the additive effects of co-activation of the repeated item remaining more active than intermediate target phonemes during recall. The step-size is limited by the repeating nature of the context-signal. For example because the period of the signal is 32 time-steps, a step-size of four would mean that after eight states of the context, the signal would repeat itself. If the step-size was very large, there would be no similarity between adjacent states of the context and co-activation of sequence items would not occur. Hence a small step-size is preferable, providing the context signal can discriminate between items of the sequence during recall.

To summarise so far, a basic model has been developed that has the temporal property that nearby states of the context signal are more similar than distant ones. Increasing the distance between successive states by using larger step-sizes during learning means sequences with repeated items can be correctly recalled.

However, the nature of speech sounds dictates that the class of each sound also has an effect on similarity, as well as simply their separation in an utterance. That is, speech sounds that share the most articulatory features are more similar than sounds that share few features. Furthermore, when more than one phoneme is involved in an error, they are more likely to be destined for similar types of position within the phonological structure (e.g. consonantal onsets). Thus structural and featural properties also play a part in producing errors, and these properties may even interact with temporal location as well. The model so far does not reflect the similarity of phonemes in terms of shared features, as each phoneme is merely represented by an individual node in the output layer, and neither does it capture phonological structure.

7.3.2 Model 7.2

The present representational scheme for phonemes can be amended to reflect the featural similarity of speech sounds by coding each phoneme as a distributed pattern of activation across a number of nodes, each of which represents an individual phonetic feature. The distinctive feature system used in the REAPECS model (Model 6.4) was again used here for Model 7.2.

Now that each phoneme has a distributed representation, coupled with the principle that nearby phonemes also activate their relevant features (which may also overlap with other phonemes), it is not a simple matter of one node winning at output. The output nodes must now be viewed as producing a pattern of activation, and the pattern of output at any one sampled time is taken to represent the phoneme with which it matches best. (The definition of 'best match' is the dealt with in the next

paragraph.) This is not an ideal way of interpreting the output, since the vocal tract moves over time. It is the trajectory through articulatory space, rather than sequences of discrete sounds that produces fluent speech. However, for the sake of investigation, a sampled representation at discrete time steps is appropriate.

The best match of the output to a phoneme was found by comparing the output against a known set of outputs in terms of how similar it is to each of them and then selecting the output with which it is most similar as the most probable output. The similarity was calculated as the cosine of the angle between the output vector and each phoneme vector (Plaut & Shallice, 1993), as in (7.6).

$$s = \frac{a \bullet b}{|a||b|} \tag{7.6}$$

where *s* is the similarity between two vectors, *a* and *b*. The numerator is the dot product between the vectors and the denominator is the product of their magnitudes. The formula in (7.6) will produce values in the range 0 to 1, the former value indicating a low similarity and the latter indicating a high similarity.

7.3.2.1 Method

The first simulation using Model 7.2 was designed to learn a simple sequence of phonemes using a distributed representation. The Model was then tested on a complex sequence.

The 16-dimensional clean context signal was again used as context with which to associate the phoneme in the sequence. This time each item or phoneme in the sequence was coded as a distributed representation. As before, the context signal was associated with each adjacent phoneme in the sequence using Hebbian learning. This time a step-size of six was used during learning, to allow greater discrimination between phonemes during recall since their representations overlapped. For reference

purposes, Figure 7.12 shows a similarity plot for the 16-element context signal with a step-size of six. When a larger step-size is used, the similarity of nearby states diminishes faster, thus after fewer states of the context signal, their similarity with each other reduces to zero. This can be seen in Figure 7.12 (cf. Figure 7.4 and Figure 7.6).

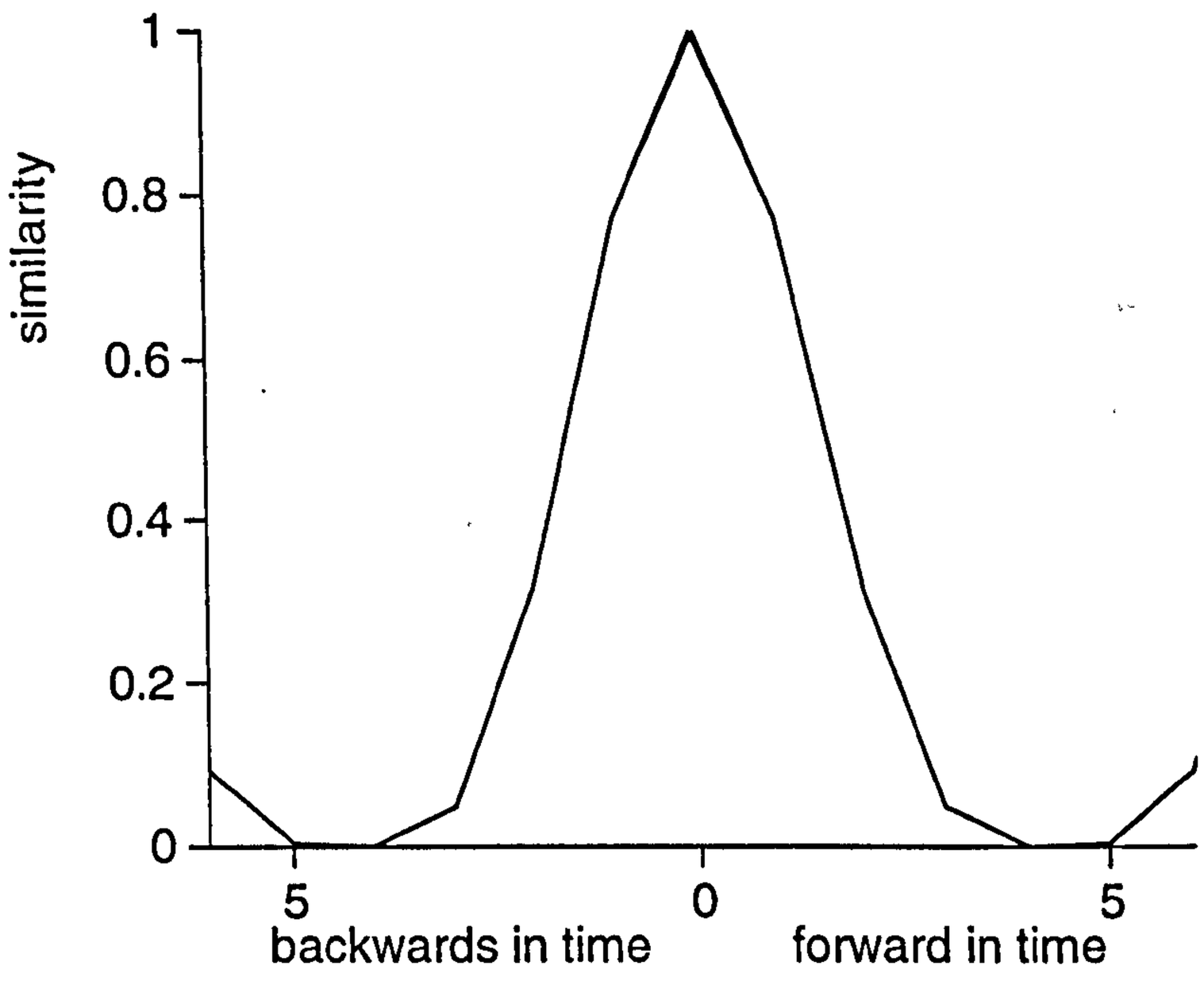

Figure 7.12. Graph showing the similarity of the context vector with a step-size of six. Similarity is plotted as ordinate against separation in successive states of the context signal as abscissa. States closer together have a higher similarity with each other.

The context signal was then reinstated to its starting position, and as it changed through time, a sequence of phonemes was recalled. Each output from the model was interpreted using the formula in (7.6).

7.3.2.2 Results

Figure 7.13 compares the output from the model at each state of the context signal to the target sequence of phonemes '/k r I s/' during recall. Each output pattern

produced by the model must be compared with a predefined set of output patterns. The result of this comparison process is a similarity value for each output pattern in the predefined set. The measure of similarity is plotted on the y-axis, against state of the context signal on the x-axis. If the most similar phoneme to the actual output is not the target phoneme and neither is it another target phoneme from elsewhere in the sequence, then it is also indicated on the graph as well.

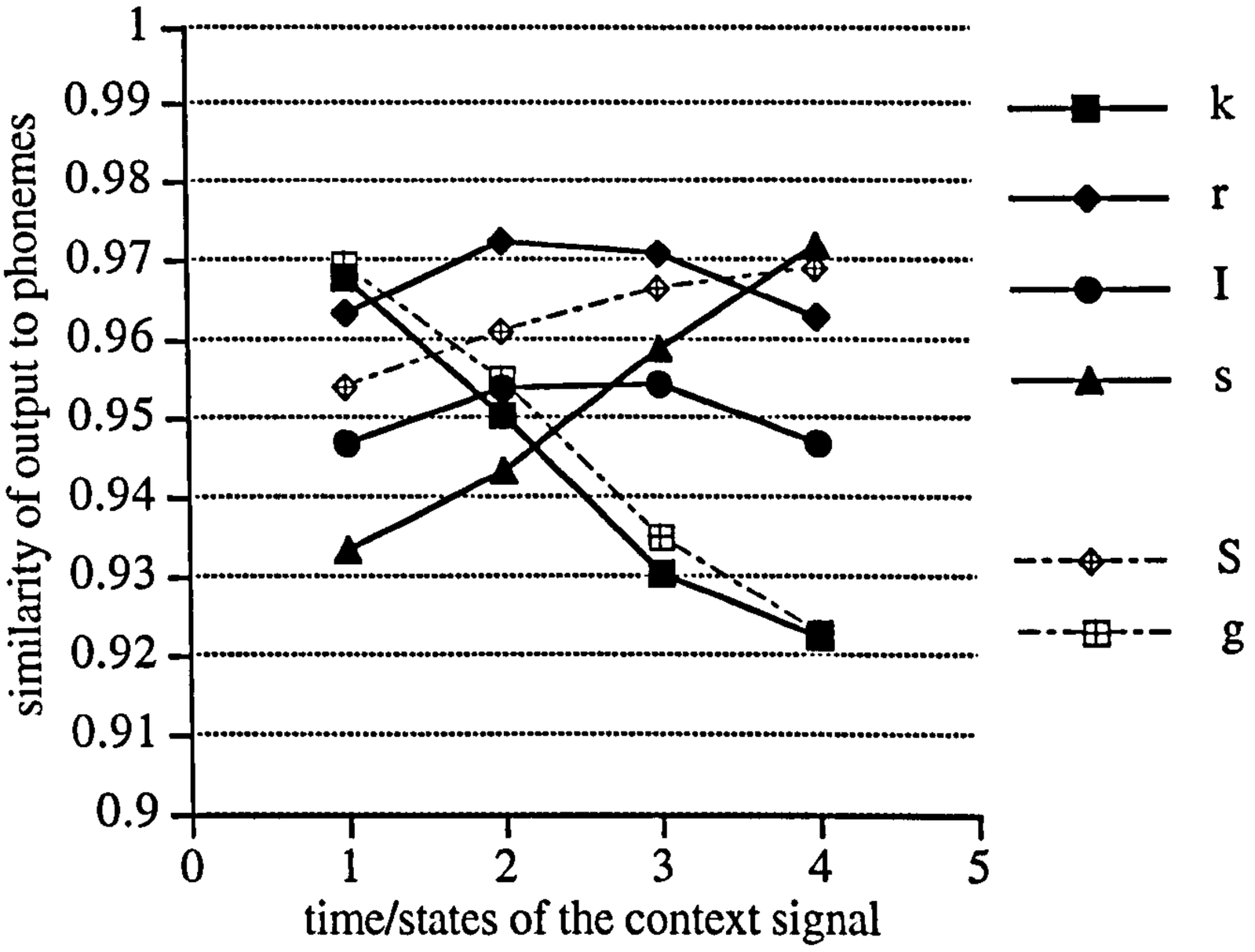

Figure 7.13. A plot of the similarity of the interpreted output from Model 7.2 with the known set of phonemes during production of the sequence '/k r I s/'.

The graph shows that the sequence recalled by the model after one episode of Hebbian learning is not the same as the intended string of phonemes. The model has produced the sequence '/g r r s/', instead of '/k r I s/'.

7.3.2.3 Discussion

The graph in Figure 7.13 shows a very different pattern of similarity (activation corresponding to allowed phonemes) for medial positions of the target sequence than for initial and final positions. The activation gradients of target phonemes in both

initial and final positions (/k/ and /s/) of the sequence are far steeper than the gradients of the medial target phonemes (/r/ and /I/). This suggests that sequence-medial target phonemes are subject to greater interference from the extra context that surrounds them (the initial and final phonemes). Sequence-initial and sequence-final phonemes only have one neighbouring phoneme to compete with, whereas medial phonemes have neighbours either side and thus will be more difficult to distinguish. Medial /r/ is also more activated than medial /I/ throughout recall; this may be because the similarity of surrounding consonants facilitates the activation of other consonants more so than vowels. The poor performance of the model may be also due to the fact that learning the sequence in just one episode of Hebbian learning with each output distributed across many units is simply insufficient to recall the correct sequence. This problem can be partly overcome by providing additional training on the target sequence.

7.3.3 Model 7.3

A simple method of accompanying Hebbian learning with additional training is to use reinforcement learning. The idea is very straightforward; when a phoneme is incorrectly recalled, the erroneous item is punished by suppressing the connections that feed into it and conversely, the connections that activate the target phoneme are reinforced. This method was used by Houghton (1990) and termed the *practice* phase in his learning procedure. The implementation of reinforcement learning in this particular instance is complicated however by the existence of negative activations from the context vector. For example, if the net input to a particular output node is to be decreased, then a parsimonious solution may simply be to decrease the strength of the connections that feed into it. However, if some of those connections are negative and the output activation from the context that flows along the connection is also negative, then a decrease in the connection strength will actually increase the net

input to the node. Hence punishment perceived as the simple idea of blindly decreasing the connection strengths could actually have an adverse effect.

7.3.3.1 Method

Model 7.3 was designed to investigate the effect of reinforcement learning on Model 7.2. The procedure for this simulation was similar to that of simulation Model 7.2, but the learning phase consisted of two phases: Hebbian association in the first phase and reinforcement learning in a subsequent phase. The reinforcement learning procedure adopted for Model 7.3 was such that punishing an incorrectly recalled phoneme involved decreasing the total net input to that phoneme node and increasing the total net input to the target one. A step-size of six was used with the context signal during both the Hebbian association phase and the reinforcement learning phase. Hebbian association occurred only once, whereas the amount of reinforcement learning in the second phase was varied. The learning rate for the Hebbian phase was set at 1, and the learning rate used in the reinforcement phase was set at 0.3.

7.3.3.2 Results

Ten simulations of the model were run, each started with a different set of oscillators to form a different context signal, to learn the sequence '/k r I s/'. All ten simulations produced the incorrect sequence after just the Hebbian learning. After the reinforcement training, the results from the simulations were mixed: seven were able to recall the correct sequence of phonemes after an average of seven episodes of reinforcement learning, the others were not able to correctly recall the sequence at all, even after many more episodes of reinforcement learning. The correct sequence was not recalled with less than six episodes by any of the simulations.

The graph in Figure 7.14 shows the results from one of the simulations that was able to correctly recall the sequence after seven episodes of reinforcement learning.

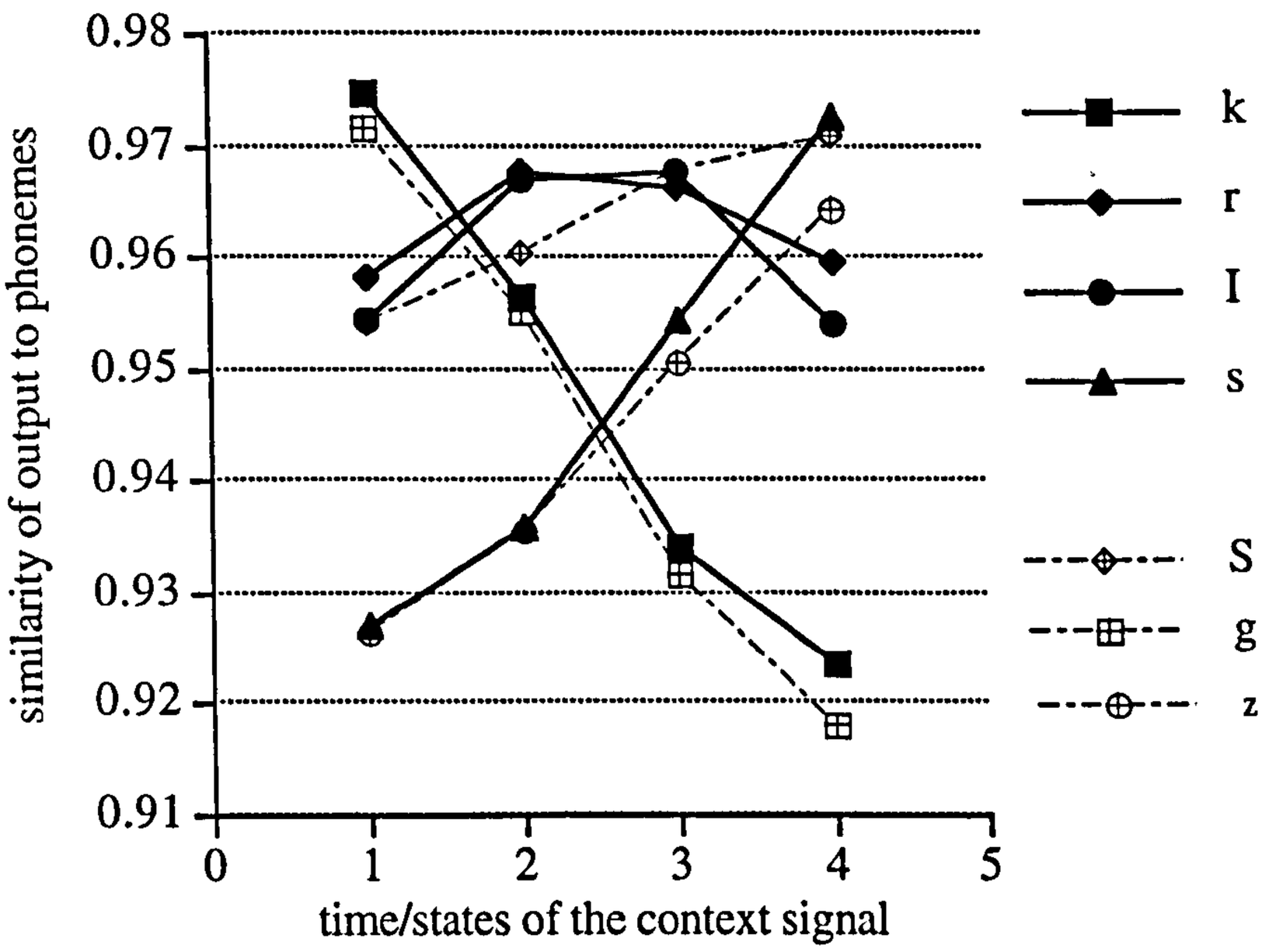

Figure 7.14. A graph showing the similarity of the interpreted model output with the known set of phonemes after seven episodes of reinforcement learning.

The graph in Figure 7.14 shows that after reinforcement learning the model is able to successfully recall the desired sequence of phonemes. The pattern of similarity of the model's output to the target phonemes throughout recall is comparable to that before the reinforcement learning, in so much that the activation gradients of sequence-initial and sequence-final target phonemes are still far steeper than those of the sequence-medial targets.

Figure 7.15 shows the results from one of the simulations which was unable to correctly recall the desired sequence, even after sixty episodes of reinforcement learning.

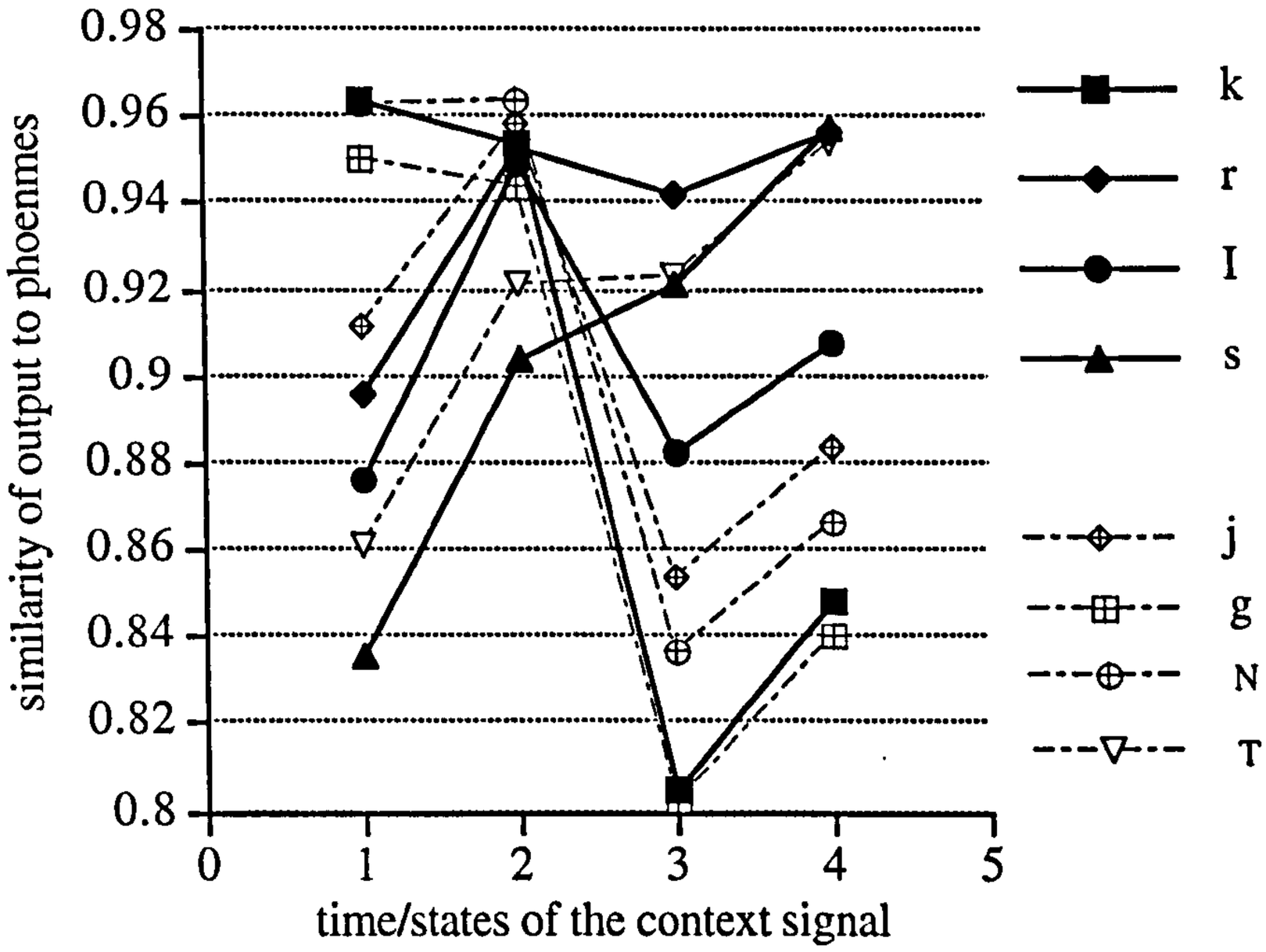

Figure 7.15. A graph of the similarity of the simulation output to that of a known set of phonemes over time for a simulation which failed to correctly recall the target sequence ('/k r I s/') after sixty episodes of reinforcement learning.

The graph in Figure 7.15 also highlights the problem of the effect that overlapping distributed representations can have on interpreting output from the model. This is particularly well exemplified by the very high similarity of the /N/ phoneme to the first target phoneme in the sequence. The correct phoneme, /k/, is more similar to the actual output of the model than /N/, but /N/ also scores highly in similarity even though it is not the most similar to /k/ in terms of shared features. Before reinforcement learning took place, the /g/ phoneme was more similar to the actual output than /k/ which appears far more plausible given that it differs from /k/ by only one feature. However, by reinforcing the representation for /k/ in the first position of the sequence, the next most similar phoneme is now /N/.

7.3.3.3 Discussion

The effect of dissimilar phonemes, such as /N/, becoming highly activated can be explained by looking ahead to the target phoneme in the next position, i.e. /r/.

Reinforcing the representation for /k/ will also reinforce (to a lesser extent) the representation for /r/ because they are close together in time and states of the context are more similar the closer they are together in time. Therefore, during recall when the initial state of the context signal is presented, the representation for /k/ will be highly activated, but the representation of /r/ will also be partially active. If we take a closer look at the featural representation of /k/, /r/ and /N/, it can be seen that the representation for /N/ shares many of the features from *both* /k/ and /r/. This is made clear in Table 7.1 below. (A listing for the abbreviated names of the features can be found in Table 6.2.)

Table 7.1.

Distinctive feature composition of phonemes which share different features which when combined closely resemble a seemingly dissimilar sound.

feature	phoneme				
	k	r	k+r	N	g
cns	+	+	+	+	+
son	-	+	+	+	-
syl	-	-	-	-	-
hi	+	-	+	+	+
lo	-	-	-	-	-
bck	+	-	+	+	+
rnd	-	-	-	-	-
int	+	-	+	+	+
str	-	-	-	-	-
nas	-	-	-	+	-
lat	-	-	-	-	-
vce	-	+	+	+	+
cor	-	+	+	-	-
ant	+	+	+	+	-

The third column of Table 7.1 shows the pattern of activation produced when all of the features for both the /k/ and /r/ phoneme are activated (at the same time). This pattern of activation, when compared with /N/, differs by two features (nasal and coronal), whereas when compared with /g/, it differs by three features (sonorant, coronal and anterior). It also differs by three features from both its constituent phonemes, /k/ and /r/, as well. Hence if all the features that make up /k/ and /r/ are

sufficiently simultaneously active at any time, then the resulting pattern of activation actually bears a closer resemblance to /N/ rather than to either /k/ or /r/, or featurally similar phonemes such as /g/.

By inspecting the activation pattern of the model in between time steps, it is possible to show the transition from articulating one phoneme to the next. In normal speech a smooth transition from one sound to the next usually occurs without any unwanted intervening sounds being articulated in between. However, Model 7.3 predicts that this is not always the case. Figure 7.16 shows the similarity of the actual output of the trained model to existing phonemes (the same as in Figure 7.14), but a more precise similarity of activation to non-target phonemes is also shown by showing the output in the time in between each state of the context signal.

If the articulatory features comprising two phonemes are active at the same time, there is a possibility that their combined pattern of activation will be more similar to a seemingly unrelated (or featurally distant) phoneme. On closer inspection of the transition from the first to the second phoneme in Figure 7.16, this certainly is the case. During the articulatory transition from /k/ to /r/, the pattern of activation from the model actually becomes more similar to the nasal /N/ than either /k/ to /r/. This also happens during the transition from the third to the fourth phoneme, where during the transition, the actual output is more similar to phoneme /S/ than either of the targets, /I/ and /s/. The combined articulatory features of /I/ and /s/ differ from /s/ by three features (sonorant, syllabic and high), from /I/ by four features (consonantal strident, coronal, and anterior), and from /S/ by three features (sonorant, syllabic and anterior). In this case the pattern for /I+s/ is featurally as close to /s/ as it is to /S/, but the particular activation produced in this instance by the model is more similar to /S/.

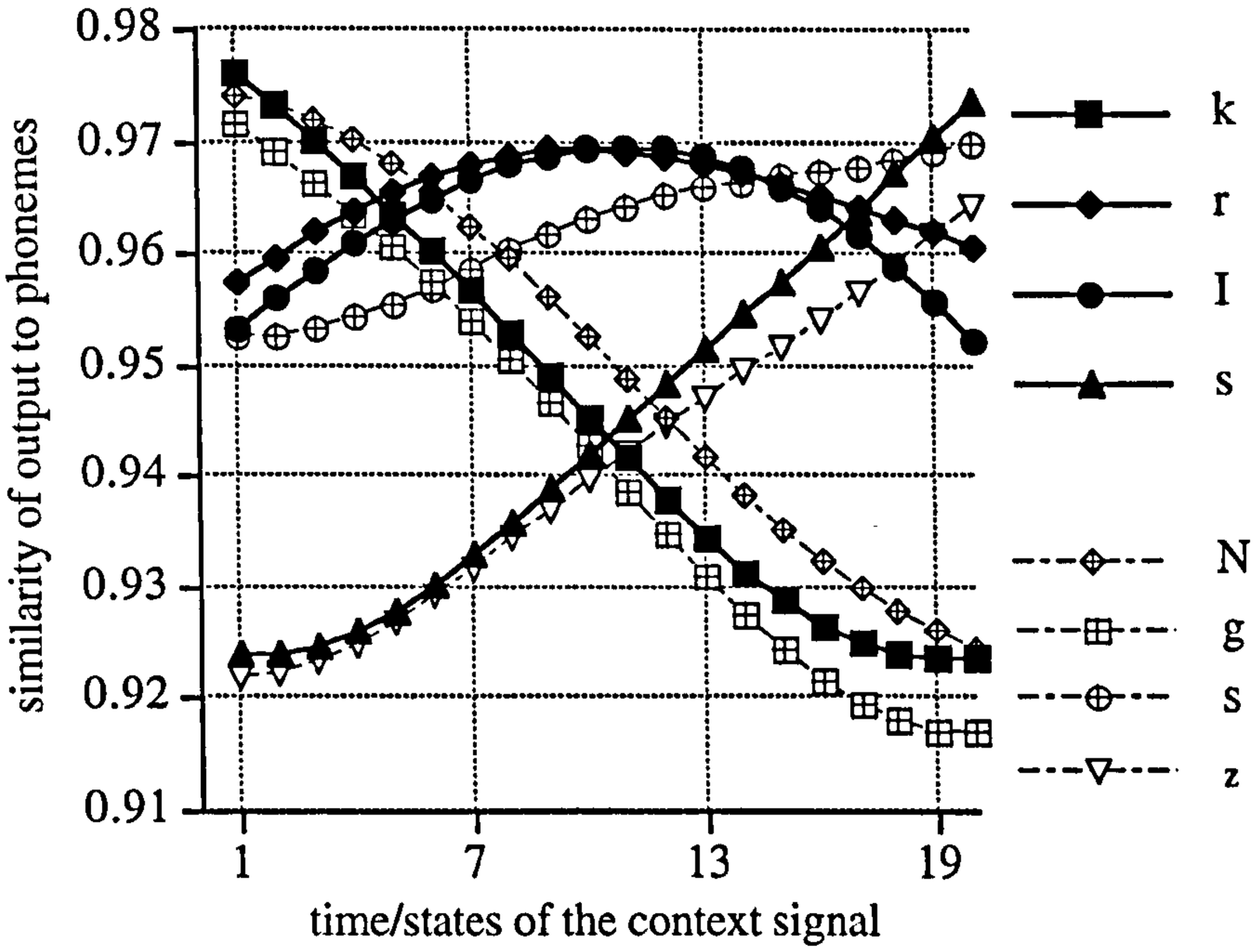

Figure 7.16. A plot of the similarity of the trained model's output with the known set of phonemes. Non-target phonemes are also plotted over time. Vertical dashed lines indicate each point in time that the output is articulated, the space in between shows the similarity to phonemes during each transition.

This could be explained by the fact that /s/ differs from /S/ by only two features (high and anterior), so when the contribution of activation to the simultaneous activation of /I+s/ becomes slightly dominated by /s/, the output is influenced by the similarity of /I+s/ to /S/ and even more so by the similarity of /s/ to /S/. It could also be explained by the time course of activation of certain features making up the /I/ and /s/ phoneme, such that some features which appear more frequently in close context will be more active and also vary less in activation than features that appear rarely.

In light of this observation, an obvious question to ask is whether this ever happens in real speech? It is not immediately apparent that such a process exists, because this could result in many speech errors violating the phonotactic rules of language, which clearly does not happen very often.

7.3.4 Summary

Model 7.1 presented the simplest implementation of an OSCAR model. Each phoneme was represented by its own dedicated output node and a 16-element context signal was associated with adjacent phonemes in a sequence using Hebbian learning. The first simulation of Model 7.1 illustrated how the model learned to recall a simple sequence of phonemes when prompted with the relevant context signal. It showed how all the phonemes contained in a sequence were active in parallel throughout the production of the sequence, with their activation influenced by the temporal relationship with other phonemes in the sequence.

Model 7.1 was also tested on its performance when presented with a complex sequence. It was found that the temporal proximity of the repeated phoneme had an additive effect during recall and that its activation was higher than any intervening phonemes when its temporal separation was small. By increasing the separation between the repeated phoneme (by using a context signal with a greater step-size) the sequence was correctly recalled.

Model 7.2 was designed to capture the featural similarities of phonemes as well as retain the temporal similarity relationship illustrated by Model 7.1. Phonemes were represented as a set of feature nodes rather than by a dedicated node and the model was first tested on its performance on learning a simple phoneme sequence. The results showed two things: first that the wrong sequence was recalled after learning, and second that featurally similar phonemes were also highly activated during recall. The activation of sequence-medial phonemes displayed a different pattern to the activation non-medial phonemes, suggesting that they may be subject to greater context effects because of their temporal position in the sequence (i.e. medial phonemes are influenced by phonemes on both sides, whereas non-medial phonemes only have one phoneme either before *or* after them). The distributed representation of the phonemes obviously interfered with learning, so a more robust

training procedure was adopted in Model 7.3.

Model 7.3 demonstrated that a simple sequence could be successfully recalled after a minimal amount of extra learning. It also revealed the unexpected property that the transition from one phoneme to the next is not always completely smooth, and that the articulatory configuration during transition sometimes bears a closer resemblance to a seemingly dissimilar phoneme than either of the targets. However, this type of error is not typically observed in human speech errors.

The context signal used so far is limited in its ability to learn complex sequences, and the length of sequence is limited by the repeating nature of the signal. In principle there is no reason why a context vector of higher dimensionality could not be used to learn longer sequences, the only proviso being to chose a vector whose elements consist of an even number of oscillators. This avoids any problems with the unlearning effect on repeated phonemes which can occur with vectors whose elements are constructed with an odd number of oscillators (as discussed earlier in section 7.2.2.4). The computational load of a context signal of higher dimensions is of course much greater. A neater solution would be to find a method for constructing a context signal with no more than 16-dimensions such that the signal was never repeated. A context signal generated in this way would be a more powerful cue for learning because longer sequences could be learned and greater discrimination between adjacent states of the signal could also be achieved.

The context signal used in the previous simulations was based on the assumption that the oscillators which make up the signal all oscillate at the same frequency but out of phase. While this type of context signal is useful for demonstrating some of the basic principles of OSCAR, it is limited in that the context signal will repeat itself after each oscillator has rotated through 360 degrees. The next section begins by describing a method for generating a non-repeating context signal and how this can be incorporated into the OSCAR model. This is achieved by

introducing a random element into the generation of the signal. In the next section I continue developing the OSCAR model, with a description of a more viable context signal.

7.4. Asynchronous Oscillator Dynamics

To generate a context signal that does not repeat, the oscillators must be made to rotate not only out of phase but also with different frequencies. One way of achieving this is to make the frequency of each *n* oscillators proportional to a power function of *n*, as in (7.7),

$$\text{freq}_n = \text{rand} \times \text{const} \times 2^n \tag{7.7}$$

where $freq_n$ is the frequency of oscillator *n*, *rand* is a random number drawn from a normal distribution of mean 0 and standard deviation 1, and *const* is a small constant, typically set to 0.00001. This method ensures that a wide range of frequencies will be generated within a range controlled by *const*.

7.4.1 The Noisy Context Signal

Context signals generated in this way, which I shall refer to as *noisy* context signals, have several attractive properties over the clean context signals described earlier. First, the signal will not repeat itself even after the oscillators have passed through 360 degrees, because they oscillate at different frequencies. So some oscillators will rotate faster than others, moving through 360 degrees before the slower oscillators. Thus by the time all of the oscillators have moved through 360 degrees, some will have moved on further than others and they will all have different values. As the context signal is continually updated, each state is therefore always different, and hence it is possible to learn much longer sequences because each

element in the sequence can be associated with a different state of the context signal. Second, noisy context signals consist of a larger proportion of high frequency oscillators due to the power function (2^n) component of the frequency generation process, as described by equation (7.7). High frequency oscillators change state rapidly and therefore differ more from time-step to time-step than low frequency oscillators. Hence the proportion of high frequency oscillators contributing to the signal will affect the distinctiveness of successive states of the signal. The adjacent states of the context will thus be more distinct from each other because there are more high frequency oscillators contributing to the signal. The random component of the generation of oscillator frequency, in equation (7.7) means that a different, non-monotonic shape for the similarity function is obtained for each set of random oscillator frequencies. An example of the similarity function of a noisy context signal is illustrated in Figure 7.17.

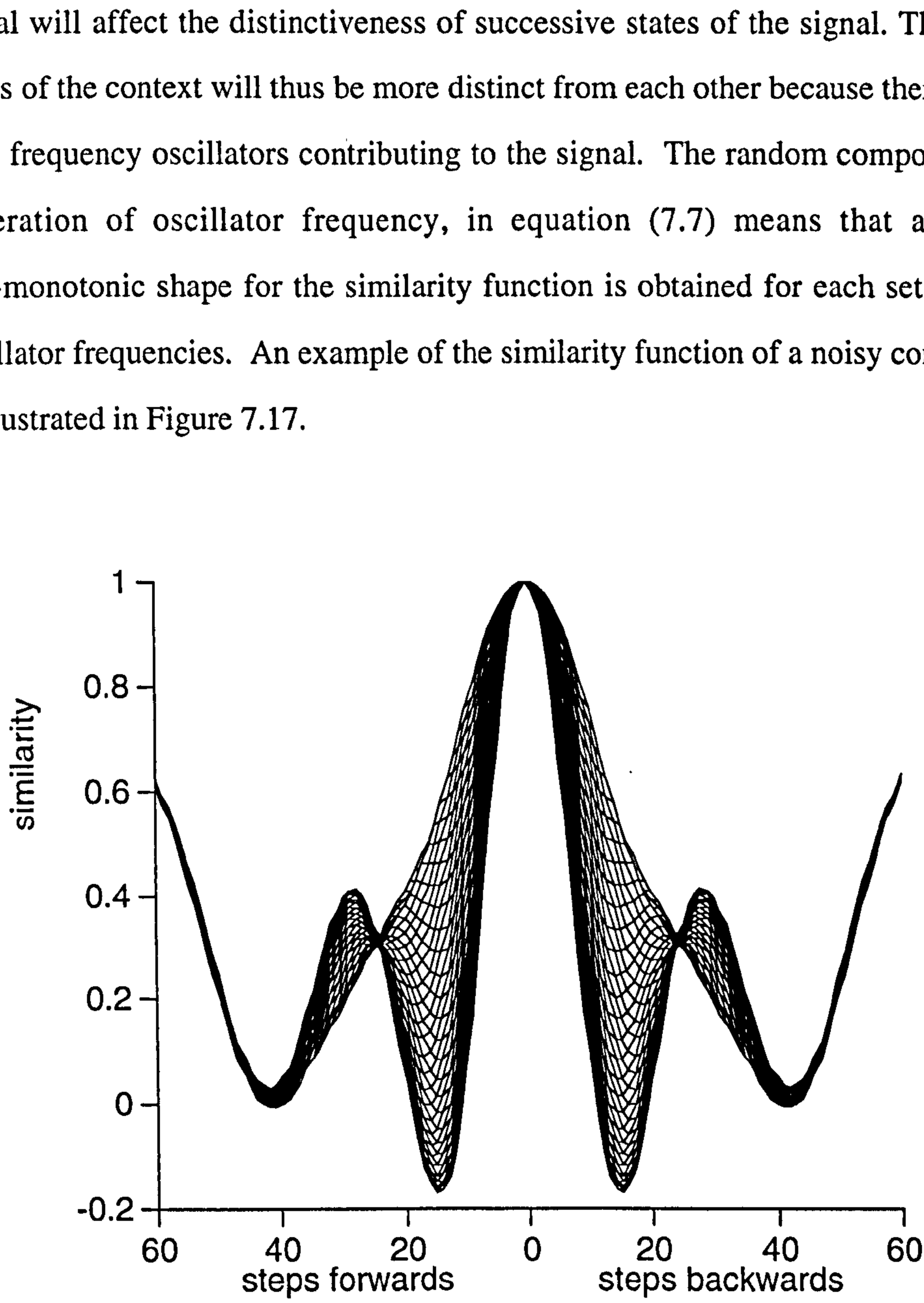

Figure 7.17. Similarity function of a noisy context signal over 64 successive states of the signal.

There is one potential problem with generating context signals in this way. The context displayed in Figure 7.17 does not show a monotonically decreasing similarity function like the clean context signal did. This can be seen by the secondary peaks in Figure 7.17 each side of the central peak. The introduction of a random element into the generation of the signal produces a non-monotonic similarity function (as in Figure 7.17), and a monotonic similarity function could be approximated by increasing the dimensionality of the signal accordingly. This could be achieved by increasing the number of elements in the context signal. This would also mean increasing the number of oscillators in the system because each element would contain more constituent parts. The dimensionality of the noisy signal has not been appropriately increased in Figure 7.17, and this could account for the non-monotonic shape of the similarity function.

7.4.2 Simulating Multi-dimensionality: A Virtual Context Signal

The disruption to the smoothness of the similarity function of the context signal in Figure 7.17 could be rectified directly by increasing the number of oscillators and hence the number of elements contributing to the signal, as suggested. However, this adds to the complexity of the signal and increases the computational load quite substantially. Alternatively, the same effect of increasing the dimensionality of the context signal could be achieved by generating many context signals and taking the average similarity function of all of them together. This latter method, which I shall refer to as creating a virtual multi-dimensional signal has been adopted henceforth for ease of computation. By computing the average similarity function of many noisy signals, the noise in the signal (as seen by the secondary peaks in Figure 7.17) can be partially removed from the resulting virtual signal and a smoother similarity function can be seen. Figure 7.18 illustrates the average similarity function of a virtual signal, obtained by averaging over 500 noisy context signals.

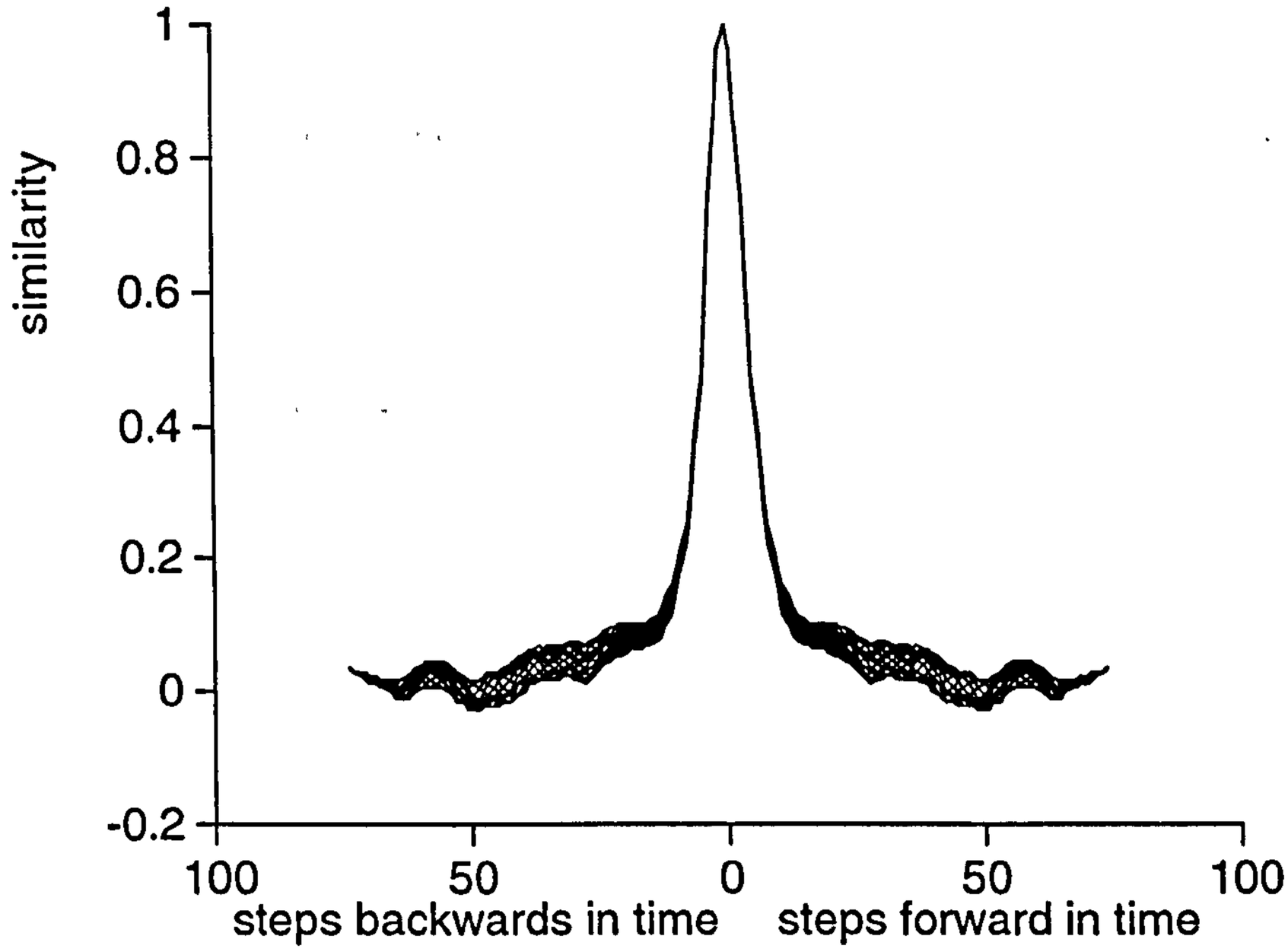

Figure 7.18. Similarity function of a virtual multi-dimensional context signal, derived by averaging over 500 noisy context signals, over 64 successive states of the virtual context.

As can be seen from Figure 7.18, the similarity function of a virtual multi-dimensional context signal produces a much smoother curve. The function still contains some noise as can be seen by the slight perturbations in the curve toward its outer edges. However, these are negligible and the similarity function plotted in Figure 7.18 is sufficiently smooth for the purpose intended.

The similarity function seen in Figure 7.18 now displays the characteristics of a context signal that could be more successfully used to learn sequential information because it does not repeat itself.

7.5. Conclusion

Simulations of the general OSCAR model confirmed the basic similarity hypothesis that temporally close states of the context signal are more similar than distant ones. In other words, phonemes closer together in a sequence were more

co-activated than phonemes far apart in a sequence. For movement errors in speech production, it is highly likely that the erroneous segments are co-represented at the time of the error. Assuming that selection is determined by the activation levels of the phoneme nodes, OSCAR then suggests that movement errors are temporally constrained, because phonemes closer together in time have much higher co-activation than distant phonemes. Thus in its simplest form, the performance of OSCAR is in accordance with findings from the speech error literature which reveal that phonemic movement errors are indeed temporally constrained. OSCAR has demonstrated a mechanism to control the parallel recall of phoneme sequences such that the co-activation levels of phonemes depended on their temporal position in the sequence, but phonemes also have a similarity measure which is determined by the features of which they are composed. This was also demonstrated by OSCAR in Model 7.2 in which a distributed representation of phonemes was used. Thus the co-activation of phonemes similar to those in the target sequence was also observed.

The OSCAR model is limited however by the nature of the context signal. In order to learn sequences with repeated phonemes, it was necessary to increase the step-size of the context signal used during learning. This limits the length of sequence that can be learnt by Model 7.1, because the clean context signal used in Models 7.1 to 7.3 periodically repeats itself in time (after 32 time-steps) and therefore has a limited temporal range. However, a method was created for constructing a context signal with a more desirable similarity function, such that states of the context did not repeat, and greater discrimination between successive states was possible. In the next chapter, the virtual multi-dimensional context signal is tested on its ability to overcome the limitations of the signal used in the models in this chapter.

Chapter 8

8. The OSCAR Model of Sound Order

The models presented in this chapter are based on the virtual multi-dimensional context signal that does not repeat itself, as described in chapter 7. The chapter proceeds with a series of simulations of the OSCAR model based on the new context signal, whose aim is to demonstrate good overall performance and account for four of the properties found in the human speech error data, as described in chapter 2. These are the basic error distribution pattern, the syllable position constraint, the distance constraint for anticipations, perseverations and exchanges, and the phonetic similarity constraint between exchanging sounds.

8.1 The Simple Model of Sound Order

In order to learn sequences using a signal with the properties of the virtual context signal, a similar virtual implementation of the process of association can be followed. This works by taking an averaged result of the recall from many noisy context signals, all of which were associated to the same phoneme sequence. This approximates the effect of using just one context signal which has a similarity function like that of the virtual context signal, in Figure 7.16. This is how learning phoneme sequences with asynchronous oscillators has been implemented for a simple OSCAR model described in this section.

The virtual context signal should now provide a more powerful cue with which to associate successive phonemes in a sequence. The noisy context discriminates

more between successive states and therefore a smaller step-size can be used to associate each phoneme in the sequence. By varying the step-size, the overall performance of the model can be controlled. The smaller the step-size, the more similar each context in which each phoneme is learnt will be and hence the harder it will be to correctly recall phonemes in their correct position. Hence overall performance should decrease as the step-size decreases and the model will be more likely to produce errors. These errors are likely to be order errors where the correct phonemes are activated but output at the wrong time. This can be demonstrated by learning a phoneme sequence with a simple model of OSCAR using the averaged results of a noisy context signal as described.

8.1.1 Method

A simple model of OSCAR using a noisy context signal was taught a mixture of simple and complex sequences by associating successive states of the context signal with successive phonemes in a sequence. For this model and the following ones the phonemes were represented by a four dimensional feature specification for the consonants and a three dimensional feature specification for the vowels, according to Wickelgren (1966, 1965), and as used in the analyses in chapter 2. Each phoneme sequence was ten phonemes long and of the form CVCVCVCVCV. Each consonant and vowel was selected randomly from the phoneme vocabulary. For each different phoneme sequence, states of the context signal were associated to each successive phoneme using Hebbian learning. The step-size of the context signal was varied. The model's response on recall was taken as the average response using 20 different noisy context signals all having been associated to the same sequence, this was to approximate the effect of a virtual context signal. Each simulation proceeded by averaging the results taken from 5000 randomly generated phoneme sequences. This

was to prevent idiosyncratic effects of noise in particular context signals from biasing the results and to produce enough data to provide a reliable result.

8.1.2 Results

8.1.2.1 Overall Performance

As expected, overall performance decreased as the step-size of the context signal decreased. Performance was best if more time (i.e. a larger step-size) was allowed to elapse in between the learning of each successive phoneme. This is well illustrated in Figure 8.1.

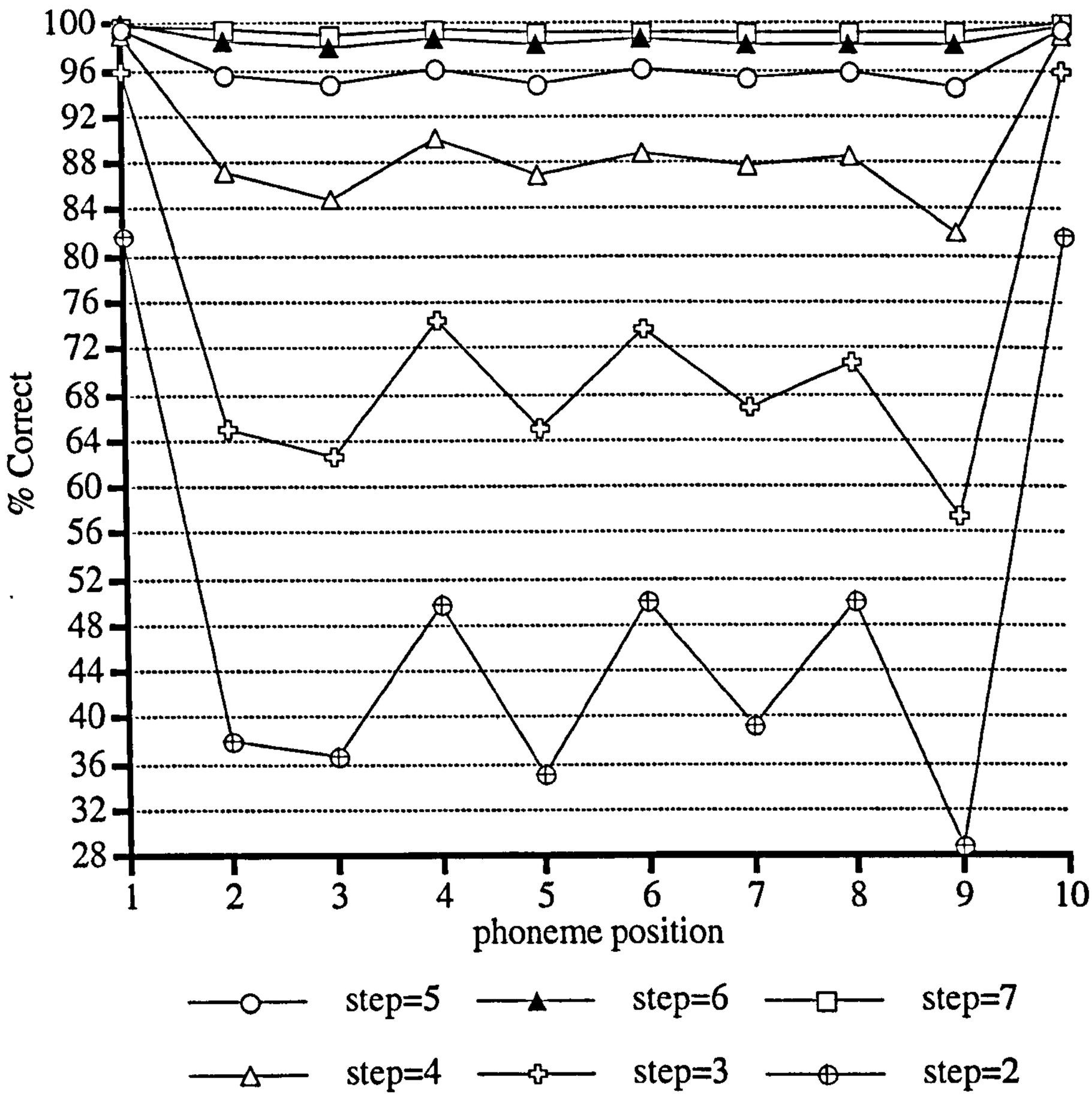

Figure 8.1. The effect of step-size on overall performance (measured as a percentage correct of all sequences) per phoneme position in a sequence (averaged over 5000 phoneme sequences).

The difference in performance depending on the time in between learning each phoneme is illustrated in the graph by different step-sizes. Small step-sizes indicate less time in between learning each phoneme, larger step-sizes indicate more time. The best performance is observed in the graph when there is more time in between learning each phoneme.

8.1.2.2 Error Type Distribution

The types of errors that the model made were categorised the same way as for the human speech error data. As the overall performance of the simple model varies with step-size, the overall number of errors increases but the rankings of error types is qualitatively the same. Table 8.1 shows the proportions of each error type as a percentage of all errors made from models with different step-sizes.

Table 8.1.

Distribution of error types (as a percentage of all errors) from a simple model associated to a ten phoneme sequence, varied over step-size. Results are calculated as the average results from 5000 randomly generated phoneme sequences, each averaged from 20 different context signals.

		error type				
step	number of errors	anticipations	perseverations	exchanges	NC subs	other
2	N=25949	22.2	17.6	0.8	36.5	22.2
3	N=13821	20.7	16.0	0.8	41.2	21.3
4	N=5446	18.3	15.1	0.5	49.5	16.6
5	N=1710	13.5	14.6	0.1	56.8	15.0
6	N=662	11.9	14.2	0	63.9	10.7
7	N=260	9.2	8.1	0	73.8	8.8

The general trend to produce more anticipations than perseverations and more perseverations than exchanges is consistent with the adult human data (Dell, Burger & Svec, 1995; Garnham et al, 1982; Harley & MacAndrew, 1995; Stemberger, 1990). Figure 8.2 gives a graphical representation of the distribution of error types from a model with a step-size of four as compared with the human data from Harley and MacAndrew's (1995) corpus.

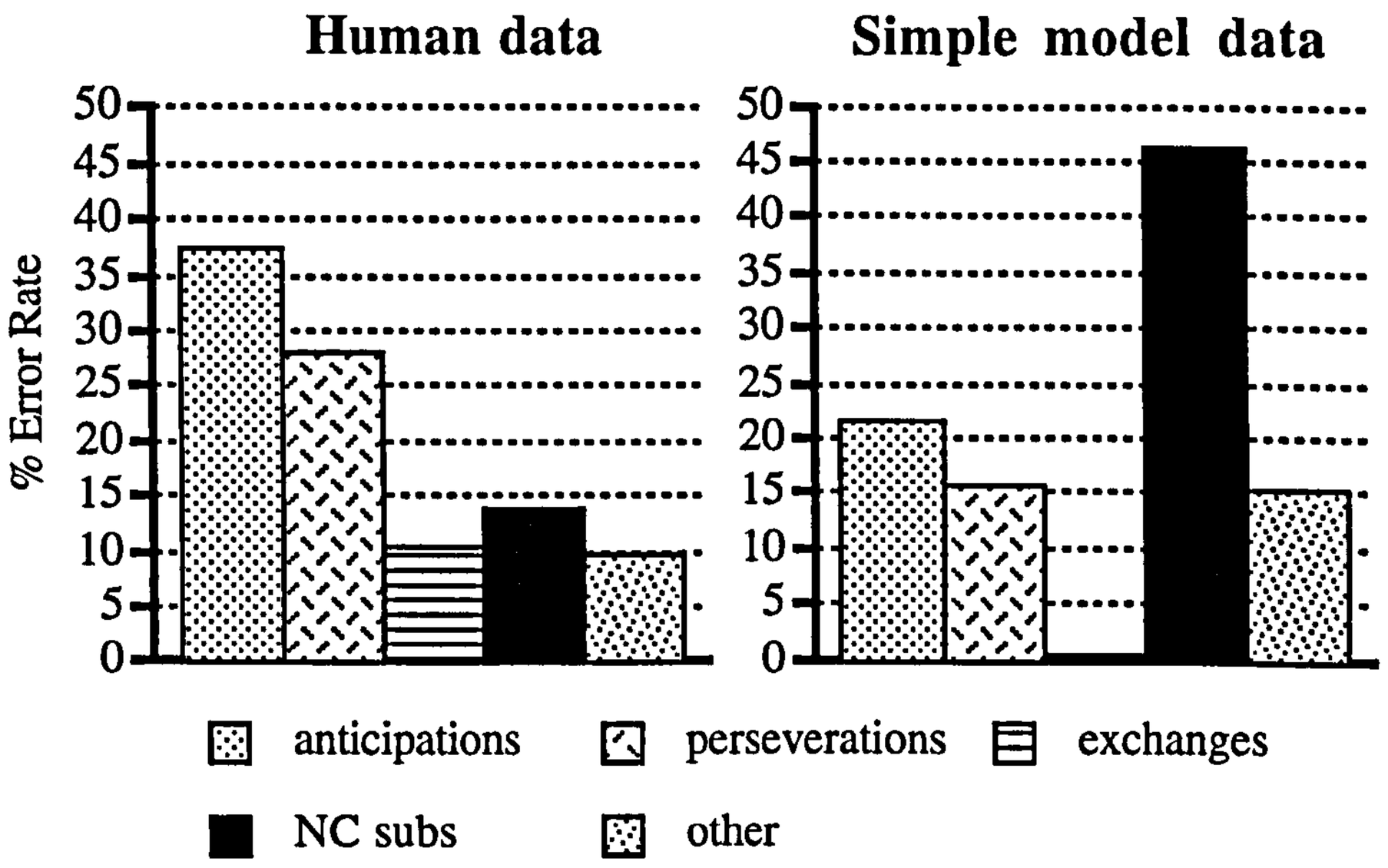

Figure 8.2. A comparison of the distribution of error types from a simple model with a step-size of four with the human data. Each error category is shown as a percentage of total errors.

The model has a strong tendency to produce a large amount of non-contextual substitution errors, and virtually no exchange errors, which is not apparent in the human data. The rest of the results in the following two sections look at the constraints discussed in chapter 2.

8.1.2.3 Syllable Constraint

The anticipation, perseveration and exchange errors produced by each model were analysed in terms of their syllabic position. Errors obeyed the syllable position

constraint if the erroneous phonemes occupied the same position within their respective syllables. As overall performance was increased with an increase in step-size, so the tendency to obey the syllabic position constraint increased. The percentage of syllable preserving errors for each error type from simulations with different step-sizes is presented in Table 8.2.

Table 8.2.

Distribution of errors within each type (as a percentage) that preserved syllabic position, from a simple model on a ten phoneme sequence for a range of step-sizes. Results are calculated as the average results from 5000 randomly generated phoneme sequences, each averaged from 20 different context signals.

	syllable position preserved?								
	anticipations			perseverations			exchanges		
step	number of errors	yes	no	number of errors	yes	no	number of errors	yes	no
2	N=5934	20.0	80.0	N=4564	20.1	79.9	N=196	3.0	97.0
3	N=2861	21.6	78.4	N=2210	19.4	80.6	N=107	0.9	99.1
4	N=978	15.9	74.1	N=804	27.1	72.9	N=26	0	100
5	N=252	37.3	62.7	N=249	47.0	53.0	N=2	0	100
6	N=79	50.7	49.3	N=94	48.9	51.1	N=0	0	0
7	N=24	58.4	41.6	N=21	71.4	28.6	N=0	0	0

Syllable position is generally not preserved in most of the errors, across nearly all of the step-sizes. The preservation of position increases as the step-size increases, in line with an increase in overall performance for each model. However, as the performance increases the number of errors on which the percentages are calculated become small, and the error pattern, as compared to the human data, is worse.

8.1.2.4 Distance Constraint

The distance each error could possibly span was limited in the model by the length of the phoneme sequence. Hence for a ten phoneme sequence of the form CVCVCVCVCV, i.e. five syllables, the maximum possible distance between error segments in terms of intervening syllables is therefore three. In order to compare the data from the model with the data from Harley and MacAndrew's (1995) corpus, the relative percentages of errors up to a four syllable separation were recalculated from the original corpus data. The recalculated distance functions for anticipations and perseverations are presented in Figure 8.3 along with the data produced from a model with a step-size of four.

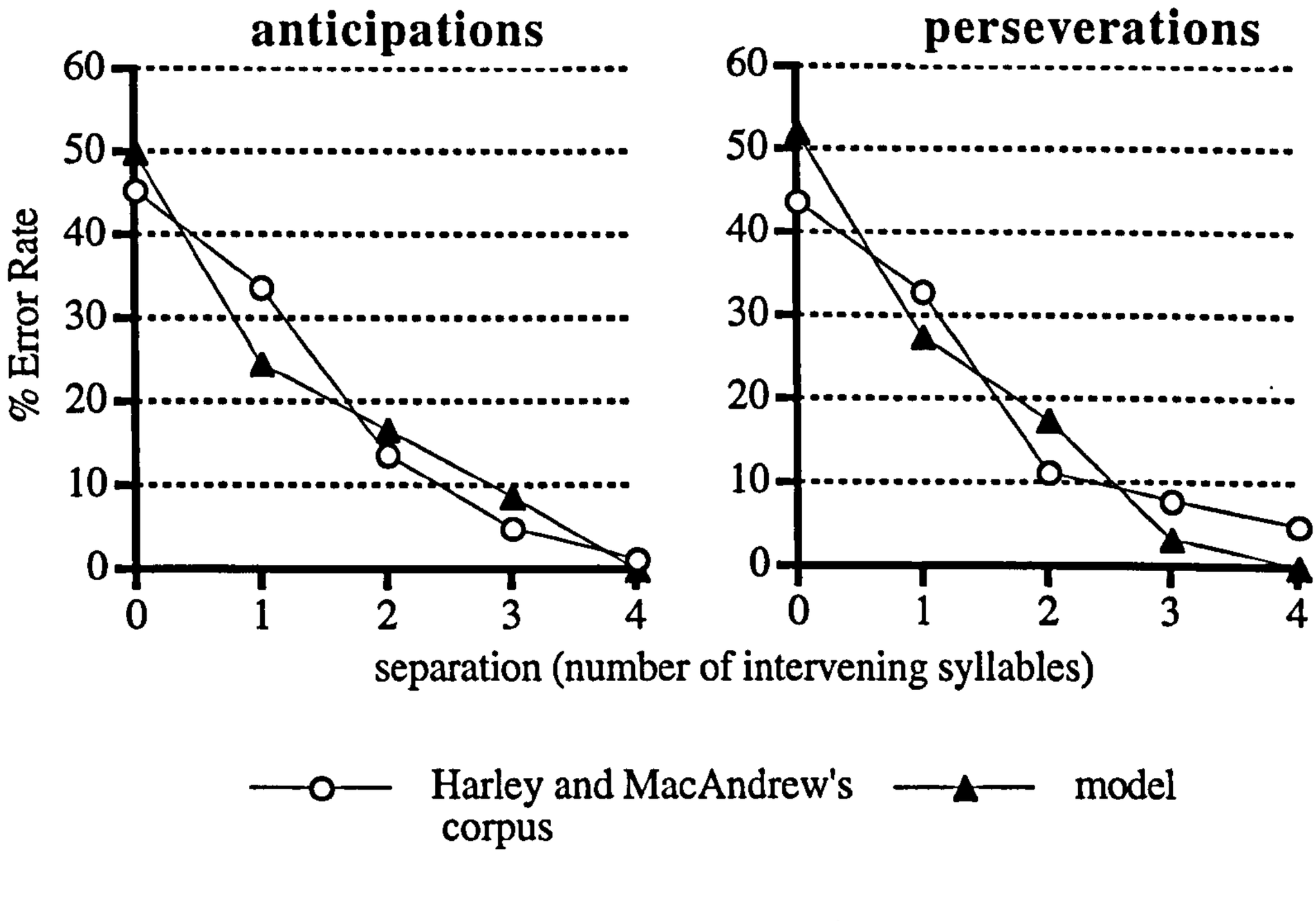

Figure 8.3. Proportions of anticipation and perseveration errors produced by a model (anticipations N= 253, perseverations N=218) with a step-size of four compared with the human data in Harley and MacAndrew's (1995) corpus for errors of up to a separation of four intervening syllables (anticipations N=820, perseverations N=591).

In both cases for anticipation and perseveration errors from Harley and MacAndrew's (1995) corpus, the number of errors with a separation greater than four syllables was small. The model failed to make enough exchange errors to present any data on the distance functions for exchanges. Both the graphs in Figure 8.3 show the basic monotonic decreasing function that is characteristic of the human data. That is errors are more likely to occur close together than far apart.

8.1.3 Discussion

The simple OSCAR model provides qualitatively a generally good level of overall performance on learning phoneme sequences. Performance varies with the amount of time that is allowed to elapse in between learning each phoneme (i.e. the step-size). Performance decreases as the step-size decreases which can be thought of as similar to increasing the speech rate. Speaking is inherently a fast and fluent process, although also prone to error. This is reflected in the model as a change in overall performance as the time in between recalling each phoneme is varied. This can be expected since a speed up in the rate of speech is likely to produce less fluent speech and hence more errors. This is certainly apparent in human speech (Dell, 1986; Dell et al., 1995). This is explained in the model by the similarity of the states of the context signal during learning and recall. When the time in between associating states of the signal to successive phonemes is small (i.e. a fast speech rate), the adjacent states of the context will be more similar to each other. Hence at any one point in time during recall, the context signal will serve to activate neighbouring and non-adjacent phonemes to a greater extent, thus increasing the likelihood of errors occurring. Speech rate also has an effect on the relative proportions of errors made. Specifically, as the rate of speech is slowed, the ratio of anticipatory to perseveratory errors increases (i.e. the proportion of anticipations increases). Dell et al. (1995) referred to error patterns with a high anticipatory

proportion (relative to perseverations) as 'good', and error patterns with a low anticipatory proportion as 'bad'. They claimed that 'good' and 'bad' error patterns could be predicted by the overall error rate, which was higher at fast speech rates. Dell (1986) showed that the number of exchange errors drops off more steeply than for other error types as the rate of speech is slowed down. The simple model also shows a drop off in the number of exchanges as the speech rate decreases, although the numbers of exchange errors are very small.

Sequences that contain repeated items are also within the capabilities of the simple model. Thus words such as "banana" are equally well produced as words such as "orange" which contain no repetition.

Of the errors produced by the model, the distribution of different types does not appear, qualitatively, to provide a good fit to the data. Specifically, the ratio of movement errors (anticipations, perseverations and exchanges) to non-movement errors (non-contextual substitutions and other errors) shows an inverse relationship to the data. That is the model produces more non-movement errors than movement errors, the reverse of which is found in the human data. This is largely accounted for by the proportion of non-contextual substitution errors produced by the model. Of the movement errors produced, the model displays the same proportional relation between them as found in the human data - more anticipations than perseverations and more perseverations than exchanges.

The syllabic position constraint is largely not obeyed by the model. This results in many errors in which a phoneme in one position in a syllable moves to a different position in another syllable. Owing to the form of the phoneme sequence being learnt (i.e. CVCVCVCVCV), this also appears as a violation of phonological class (i.e. consonant or vowel). That is all the errors that violate the syllabic position constraint also change their phonological class when they appear as an error. This clearly contradicts the observations from human speech errors. However, phonological class

would no longer be violated if the syllabic position constraint was obeyed in the model. Furthermore, the model cannot account for the ratio of consonant to vowel errors seen in the data. The model generally produces more vowel errors that consonant errors. This could be influenced by pattern of syllabic violation errors, and further discussion will be postponed until the syllabic position violation errors are reduced to a more realistic level.

Of the errors that do obey syllabic position constraints, the shapes of the distance functions over which the errors span provide a good qualitative fit to the human data. Errors are more likely to occur between nearby sounds rather than far apart ones. This effect is accounted for in the model primarily by the nature of the context signal in which similarity of state is a function of the temporal position of the signal. Therefore phonemes produced near to each other in speech are represented by the similarity of the context signal associated with those phonemes. Since the context of nearby phonemes is similar their co-activation is brought about and hence the likelihood of a nearby phoneme being inappropriately selected is raised.

A simple OSCAR model has already shown how a serial order mechanism could work within a model of speech production. Sequences of phonemes, even with repeated phonemes within them, are associated to successive states of a temporally changing context signal and successfully recounted on recall. Of the errors involving a serial order component, the errors display a distance constraint very similar to that observed in the human data. However, there are several prominent issues unaccounted for by the model. These are the lack of exchange errors, the over-abundance of non-contextual substitution errors and the disregard for the syllable constraint in the errors. These shortcomings motivate some modifications to the simple model, presented in the next section.

8.2 The Synchrony Model

As it stands, the OSCAR model lacks an adequate account of the syllable position constraint. In this section I explain how a simple timing mechanism can be added to the model to account for position constraints within a syllable. There is evidence from research into the perception of syllable timing in speech that production is in some sense rhythmical (Fowler, 1979, 1983). Simply put, this means that speech can be measured, and is perceived to occur with regular intervals. Whether or not the regular intervals, or rhythmic cycles, occur at the onset of a particular unit in speech (Ferreira, 1993), or are based on the perceived timing of stressed vowel production (Fowler, 1983; see also Meyer, 1994), is not crucial here. The important point is that there exists some kind of mechanism whose task it is to produce bursts of speech at synchronous intervals as part of the production process. But if this kind of synchronous production does exist in speech, then where does this leave the model? What is required is an additional component in the context signal that repeats in synchrony with the rhythmical production of speech. This should result in the contextual cue for phonemes within a syllable being more similar for nearby phonemes in the same syllabic position than for nearby phonemes in different positions within the syllable. The next simulation demonstrates how synchrony can be included in the model by a simple modification to the context signal.

8.2.1 Method

The context signal introduced in this chapter was designed so that the signal should not repeat itself, so that states in the signal should not recur. What now seems to be necessary to account for the rhythmic properties of speech is a combination of both a repeating and a non-repeating combination of context signals. What is needed is a component of the context signal that does actually repeat itself so that it is similar each time a syllable is encountered. This can be demonstrated by superimposing a

clean, repeating context signal on the virtual, non-repeating multi-dimensional context signal used in the simple model. Recall that the way the virtual multi-dimensional signal was implemented was by taking an averaged response from many 16-dimensional noisy signals. All that needs to be changed to include a timing element is simply substituting some of the noisy signals for clean, repeating ones. Specifically, the similarity of a synchronous signal is calculated by taking the average dot product (the similarity measure) of say 15 clean, and five noisy context signals for each state of the 15 clean and five noisy signals. The effect on the similarity function of the new synchronous context signal, calculated in this way is shown in Figure 8.4.

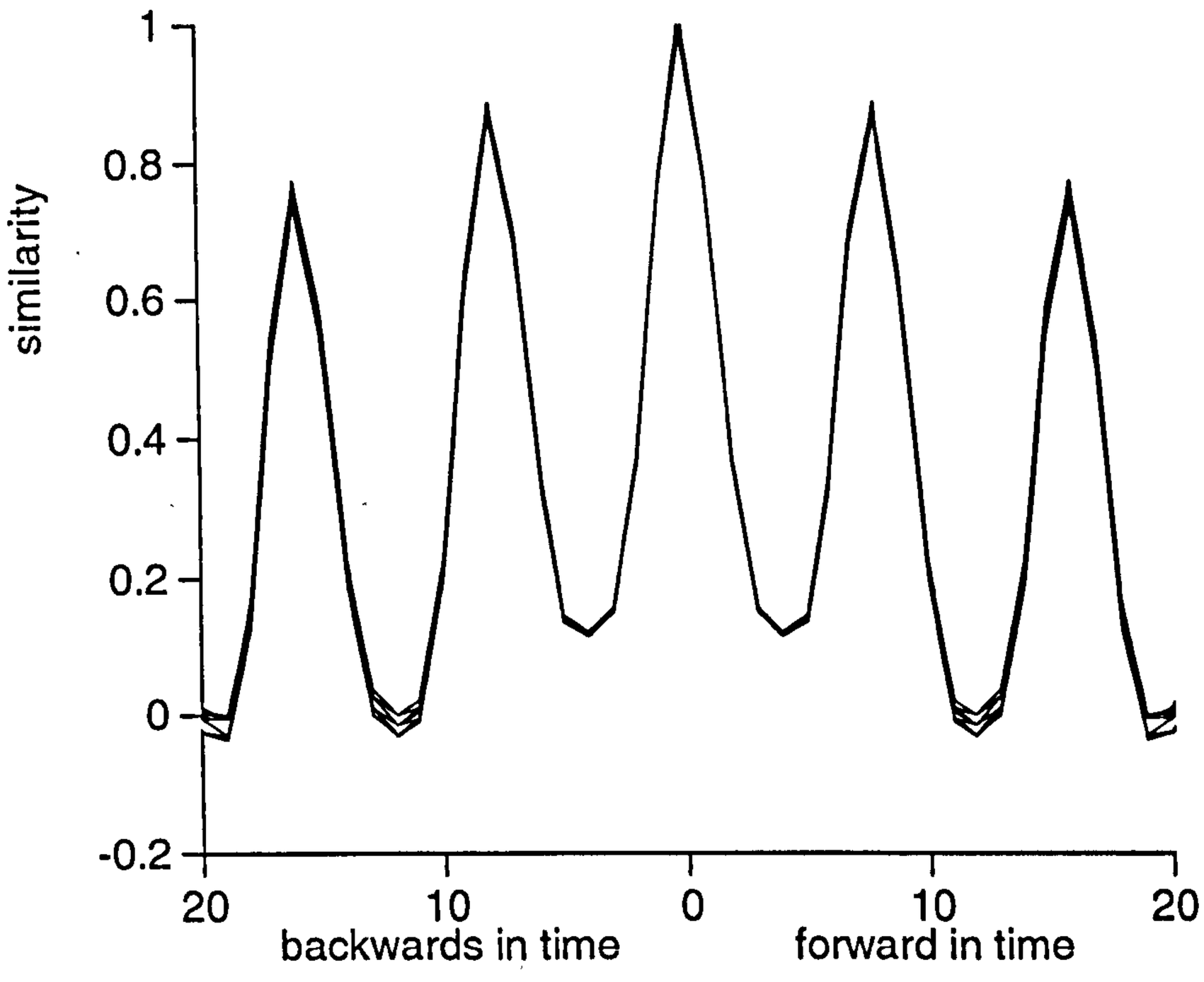

Figure 8.4. A plot of the similarity function of the virtual multi-dimensional context signal with a timing element included such that 75% of the virtual signal is made up from a clean, repeating signal. The peaks either side of the central peak indicate the presence of the timing element in the signal.

The general similarity principle is still obeyed by the new context signal in that the most similar state to the current state is itself, with a decrease in similarity for

more distant states. However, peaks of similarity occur either side of the current state. These are due to the inclusions of the clean, repeating context signal, and relate to the timing element of the new signal. By associating each peak with adjacent syllables, each phoneme gains some within syllable identity because now not only will temporally close phonemes be more similar to each other, but their similarity will also be affected by their position within the syllable. Homogeneous syllable constituents will be more similar than heterogeneous constituents, and thus syllable-initial phonemes will be more similar to each other than say syllable-initial and syllable-final phonemes. Therefore the errors made by the model should tend to obey the syllable position constraint providing that the synchrony in the context signal is strong enough.

Five-syllable phoneme sequences, generated in the same way as for the simple model, were associated to the synchronous context signal. The strength of the synchrony in the context signal was varied by systematically increasing the proportion of the virtual signal that was made up from the clean, repeating signal. A step-size of four was used for the context signal. For each simulation of different synchrony strength, the model was associated to 5000 different phoneme sequences and overall performance and error classification were based on the level of performance achieved during the recall of each of the 5000 sequences.

8.2.2 Results

8.2.2.1 Overall Performance

The effect of introducing synchrony into the context signal on the overall performance of the model is displayed in Figure 8.5. A high level of performance is still achieved when the synchrony is weak, but gradually decreases as the synchrony increases. This is because as the strength of the synchrony increases syllable

homogenous phonemes become more similar and hence are more likely to be confused.

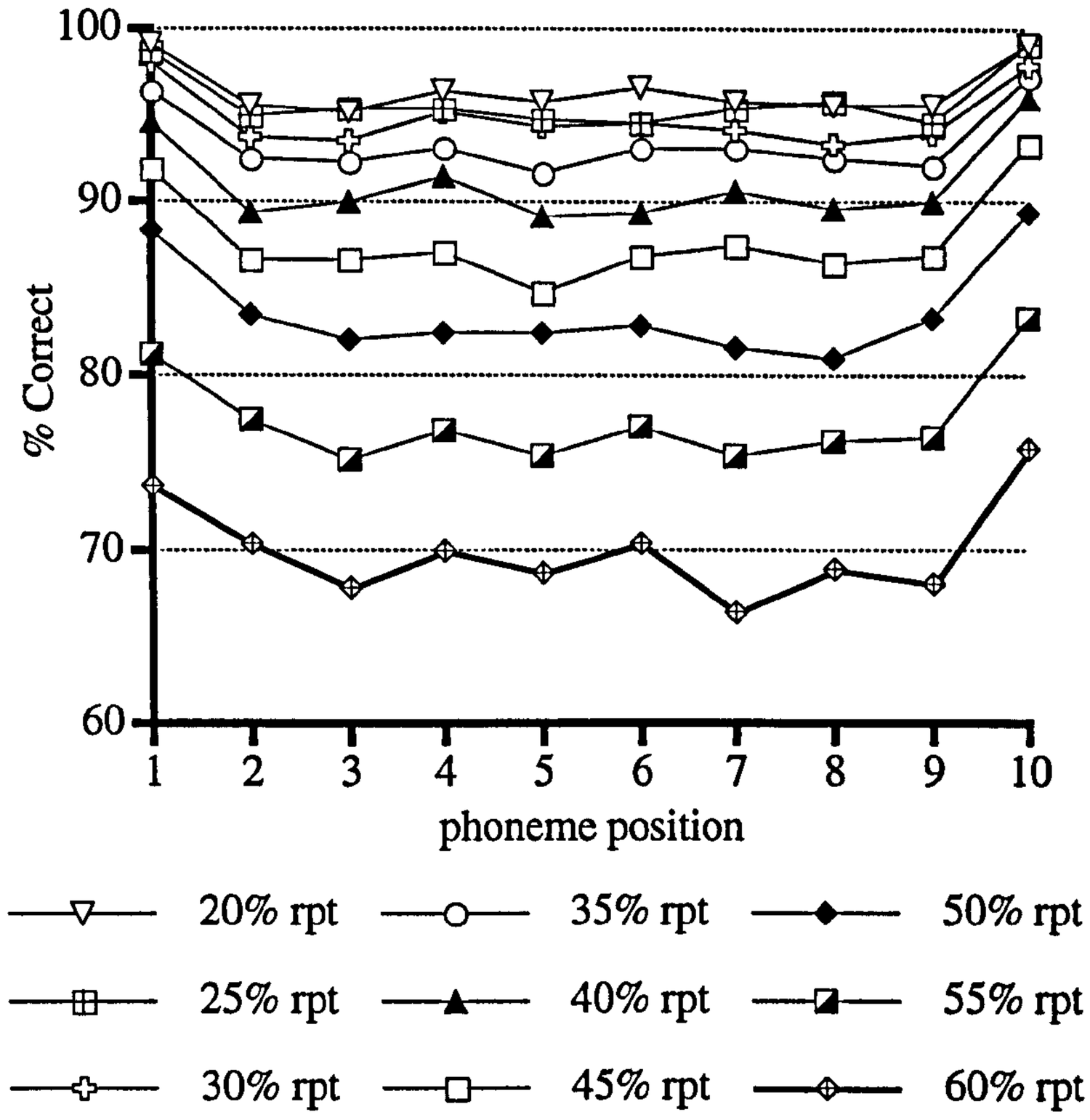

Figure 8.5. The effect of the strength of the synchrony introduced into the context signal on the overall performance of the model. Each line in the graph represents the overall performance from a model with a fixed proportion of synchrony (in %rpt, or the percentage of the signal which consisted of the clean, repeating signal) in the virtual context signal with which the phonemes were associated.

8.2.2.2 Error Type Distribution

The errors made by the model were categorised in the same way as for the simple model. Table 8.3 shows the distribution of error types (as a percentage of total errors) for each simulation of the model with a fixed amount of synchrony, varying from 5% to 50%. For comparison, the results from the simple model with 0% synchrony are also included in the table.

Table 8.3.

Distribution of error types from the synchrony model (as a percentage of total errors).

percentage of signal consisting of clean signals	number of errors	% anticipations	% perseverations	% exchanges	% NC subs	% others
0	N=5446	18.3	15.1	0.5	49.5	16.6
5	N=4320	17.3	14	0.1	54.6	14.1
10	N=3753	16.4	12.3	0.05	60.4	10.9
15	N=3705	14.1	11.8	0.02	65.8	8.3
20	N=3713	12.9	12.8	0.02	68.8	5.5
25	N=4140	15.1	13.6	0	67.3	4.0
30	N=4735	16.6	13.8	0.02	66.1	3.5
35	N=5563	18.9	12.6	0.01	64.2	4.4
40	N=6979	18.9	13.8	0.01	60.8	6.5
45	N=8636	20.5	15.0	0.02	56.7	7.8
50	N=11081	23.8	17.6	0.01	50.8	7.8

The synchronous extension to the model has had little effect on the qualitative pattern of errors in the model. That is, there are still more anticipations than perseverations and more perseverations than exchanges. Furthermore, there are still more non-contextual substitution errors than any other type of error. Figure 8.6 shows how the error distribution from a model with 45% synchrony in the context signal compares with the distribution of errors found in Harley and MacAndrew's (1995) corpus.

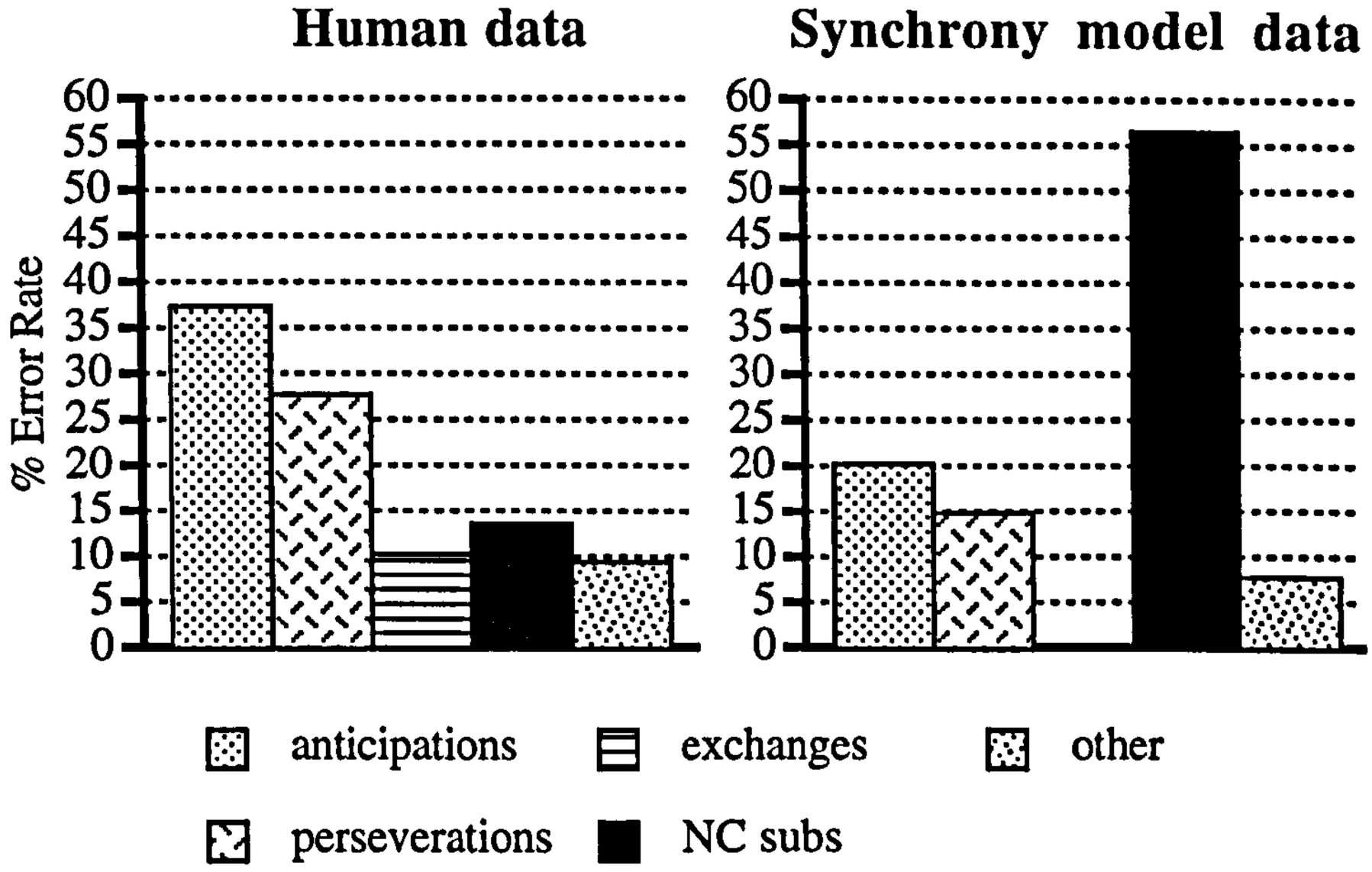

Figure 8.6. Comparison of distribution of error types from a 45% synchrony model with the human data.

8.2.2.3 Syllable Constraint

The effect of introducing synchrony into the context signal was predicted to affect the nature of the movement errors in terms of their adherence to the syllabic position constraint. Specifically it was predicted that by introducing a synchronous mechanism into the model, the number of errors violating the constraint would fall as the level of synchrony was increased. The outcome of this prediction can be seen in Table 8.4.

The result is quite clear. As the strength of the synchrony in the context signal changes (in %) so does the ratio of syllable-preserving to non-preserving errors. As synchrony increases, the number of errors that violate the syllabic position constraint falls. This effect is illustrated in Figure 8.7 which shows the relationship between the proportion of errors that violate the constraint and the proportion of synchrony present in the context signal of the model.

Table 8.4.

Distribution of errors, within type, observing the syllable position constraint (as a percentage within each error type), from the average results from a synchrony model trained on 5000 phoneme sequences.

		syllable position preserved?				
		anticipations		perseverations		
percentage of signal consisting of clean signals	number of errors	yes	no	number of errors	yes	no
0	N=978	36.9	74.1	N=804	27.1	72.9
5	N=747	44.7	55.3	N=603	41.5	58.5
10	N=615	56.7	43.3	N=461	60.1	39.9
15	N=524	73.5	26.5	N=437	77.1	22.9
20	N=479	87.4	12.6	N=473	85.6	14.4
25	N=626	96.2	3.8	N=563	95.0	5.0
30	N=784	97.7	2.3	N=653	97.7	2.3
35	N=1049	99.7	0.3	N=699	99.7	0.3
40	N=1319	99.9	0.1	N=961	99.5	0.5
45	N=1769	100	0	N=1296	100	0
50	N=2638	100	0	N=1950	100	0

Without any synchrony in the context signal, both anticipation and perseveration errors are mostly ones that violate their syllabic position when the sounds move. As synchrony approaches 50% in the context signal, the constraint violating errors disappear.

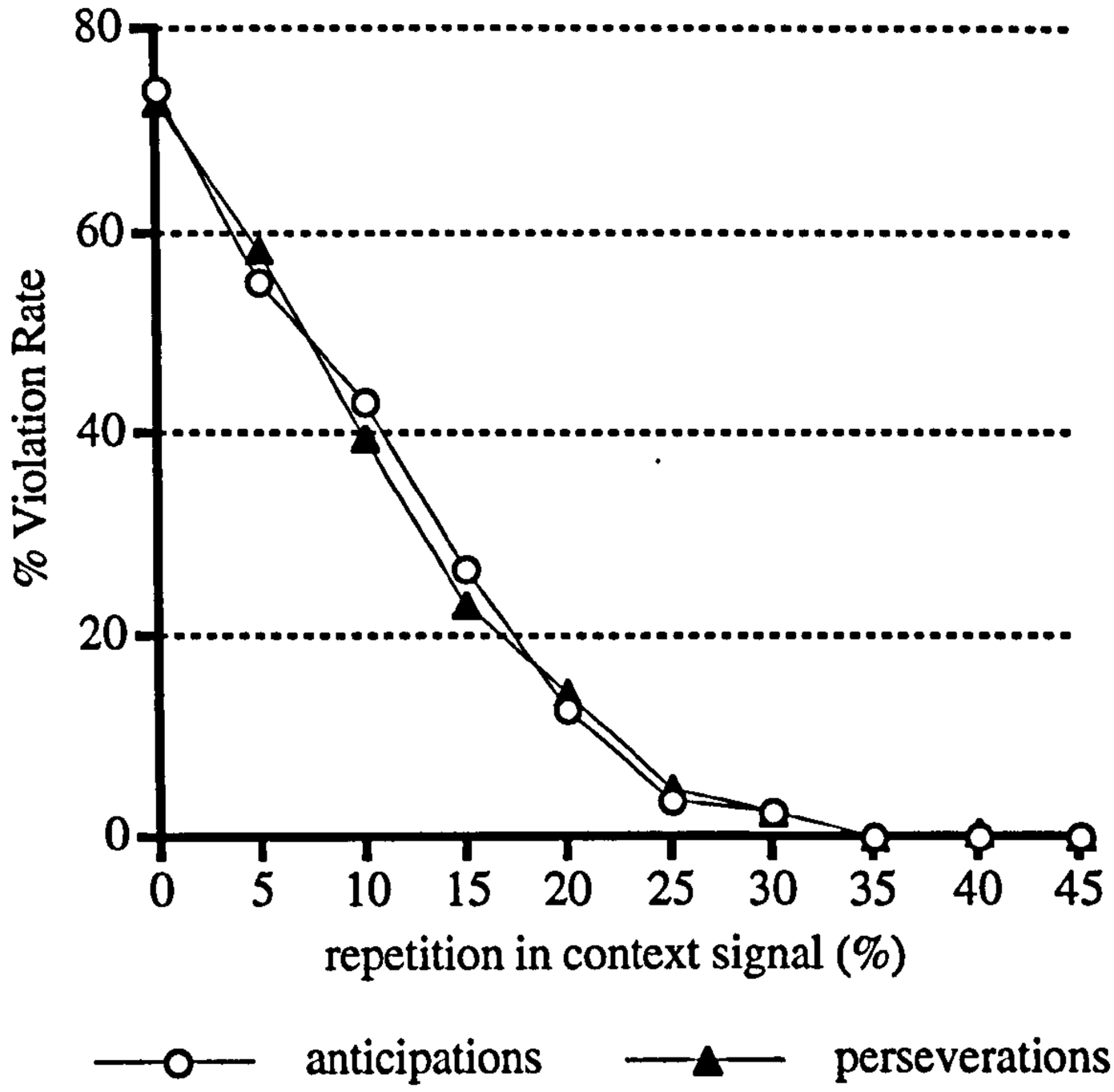

Figure 8.7. Rate of decrease in syllable-position violating anticipation and perseveration errors from as a function of the proportion of synchrony in the context signal.

8.2.2.4 Distance Constraint

The distance functions for anticipations and perseverations (collapsed over consonants and vowels) are shown in Figure 8.8. As for the simple model, the data from Harley and MacAndrew's corpus have been recalculated for errors up to a four syllable separation for comparison with the model. Again, there were too few exchange errors to present any analyses.

The distance functions for both anticipation and perseveration errors are qualitatively preserved. That is the presence of synchrony has not affected the relative proportions of the number of syllables across which error segments span, and errors are more likely to occur close together than far apart.

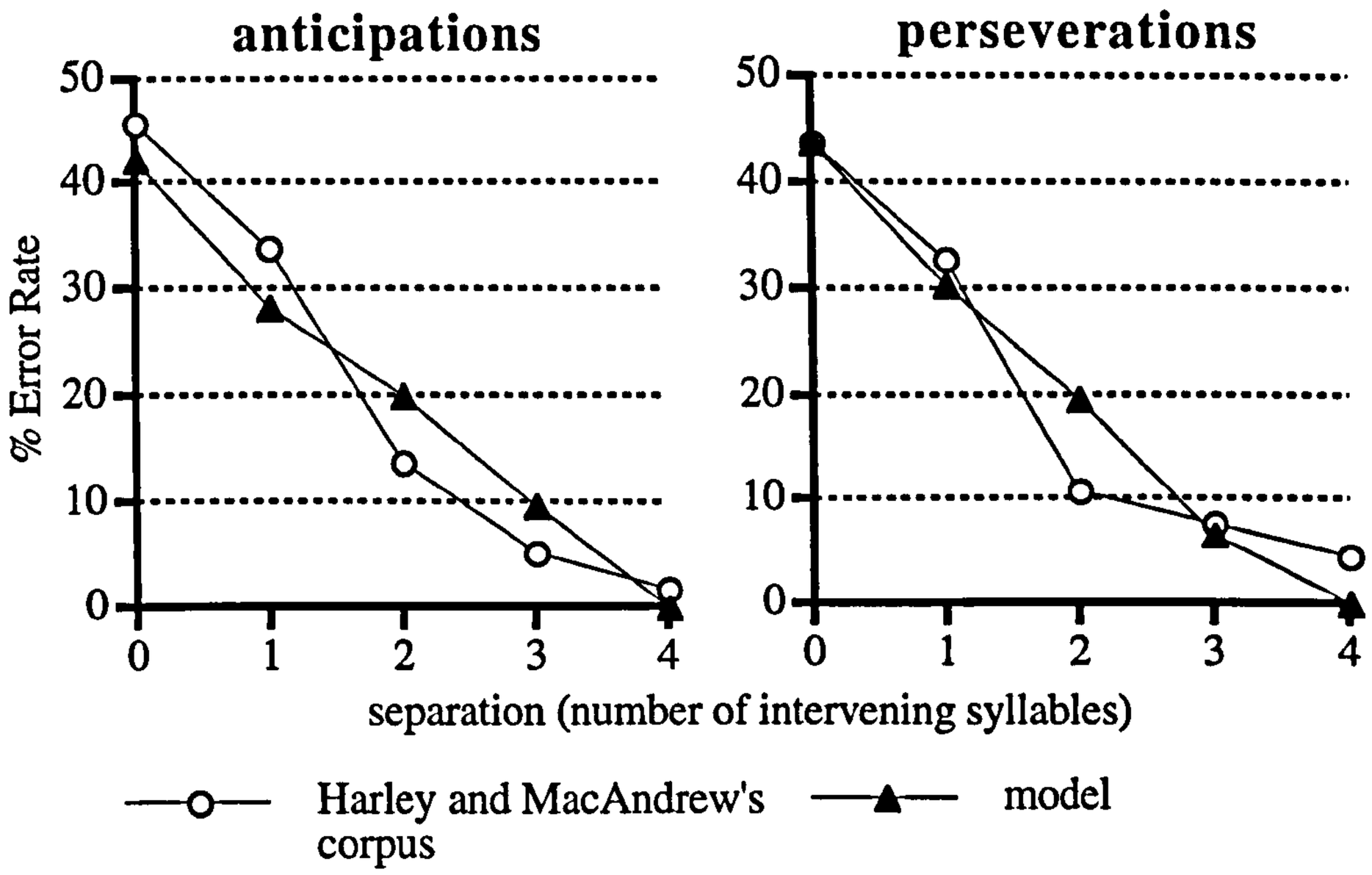

Figure 8.8. Proportions of anticipation and perseveration errors produced by a synchrony model with 45% synchrony in the context signal (anticipations N=1796, perseverations N=1296) compared with data from Harley and MacAndrew's (1984, 1993) corpus for errors up to a separation of four syllables (anticipations N=820, perseveration N=591).

8.2.3 Discussion

The introduction of a synchronous mechanism into the model has had the predicted effect of minimising errors that violate the syllable position constraint. It has had little effect on the distribution of error types produced by the model or the distance gradients for the errors it produces.

The synchrony in the context signal is provided as a purely instrumental mechanism to show the effect of a temporal approach to accounting for the syllable position effect without recourse to a syllabic frame. This has been demonstrated by introducing into the context signal a timing element that synchronises the production of each syllable. This synchronism does not make any claims as to the place of synchrony within the syllable, only that the production of speech is indeed rhythmical and synchronised in some way. The synchrony model can now account for the

syllabic position constraint with a temporal process. However, the non-existence of exchange errors and the large proportion of non-contextual substitution errors still remains problematic for the model. Given that the model does produce anticipation and perseveration errors and that an exchange error involves an anticipation and then a perseveration of the sound dislocated by the anticipation, it is plausible that the model can be extended to account for these data. These problems are addressed in the next section, which introduces a mechanism for "switching off" a sound once it has been produced so that it is less likely to be articulated again in the immediate future.

8.3 The Inhibition Model

It seems that the synchrony model provides a good account of the syllable position constraint, but a major weakness is its failure to account for why exchange errors occur. In this section I first introduce some extensions to the model, motivated from both psycholinguistics and serial order in short-term memory which are then implemented in the new inhibition model. The aim of the inhibition model is to provide an account of the relative proportion and properties of exchange errors found in the human data. The results presented for this model will therefore focus on exchange errors.

8.3.1 Inhibitory Processing

Inhibition, the process that effectively "switches off" or prevents the activation of an item once is has been produced, is evident both within the literature on speech production (Dell and O'Seaghdha, 1993; Houghton, 1990; Sevald & Dell, 1994), spelling (Houghton et al., 1994), selective attention (Houghton, 1992; Houghton & Tipper, 1994), and short-term memory (e.g. Page & Norris, 1995). Inhibitory processes are important in short-term memory because once an item in a list has been recalled it is very unlikely to be recalled again. Therefore once an item has been

recalled, it can be almost completely inhibited. The concept is not so straightforward in speech, because phonemes are often repeated in the same word or utterance and hence obviously not totally inhibited after they are output. The inhibition must more likely be partial, allowing phonemes that are intentionally repeated to be recalled.

This view of inhibition is compatible with the synchrony model. In the model, an intentionally repeated phoneme receives independent activation from the temporally changing context signal when the time the second occurrence is planned approaches. Partially inhibiting the first occurrence of the repeated phoneme would not necessarily prevent recall of the second phoneme because independent activation is provided by the context signal. However, unless a phoneme is *planned* to appear more than once, then inhibition should affect the unwanted duplication of phonemes, like anticipation and perseveration errors. Specifically, inhibition would reduce the number of perseveratory errors. However, inhibition could adversely affect the occurrence of anticipation errors. An anticipated phoneme once inhibited would have a reduced chance of the anticipation being completed by making the source of the anticipation a less likely candidate and clearing the way for competitor phonemes to be output instead. If the nearest competitor to the now inhibited source was the phoneme displaced by the locus of the anticipation, then an exchange error would occur. In the present model, co-activation of phonemes is symmetrical around the current phoneme, thus the displaced (and unarticulated) phoneme has no special advantage over upcoming phonemes that are equidistant (in time) from the source of the anticipation. It is therefore still not clear as to how inhibitory processes could play a role in accounting for the production of exchange errors unless there is another process at work that facilitates the output of phonemes if they "miss" being output in their intended place in the sequence. Evidence of this sort of process can be found in studies of short-term memory and spelling (Houghton et al., 1994).

8.3.2 The Persistency Effect

Recently, it has become apparent from the short-term memory literature that transposition errors (the short-term memory terminology for exchange errors) are subject to a property called *fill-in* (Page & Norris, 1995). This can be stated as the tendency for an item in a list, should it not be recalled in its correct position, to be most likely recalled in the next position. This property predisposes serial order systems to make transposition errors that are mainly one-apart. The same property is also obviously observed in the speech error data. When phonemes exchange they tend to be close together rather than far apart. Thus if a phoneme is displaced by another phoneme from later in the sequence (i.e. an anticipation occurs), then the displaced phoneme, if it is going to be output at all, is more likely to be output sooner rather than later, thus resulting in an exchange of phonemes over a shorter rather than a longer distance. I will refer to the fill-in property found in speech error data as the *persistency effect.*

The persistency effect in speech production is incompatible with chaining models because a chaining account would predict that should an item be recalled prematurely then the next item to be recalled would most likely be the next item in the list rather than the omitted target. Persistency is not an emergent property of the current OSCAR model because the association between the context signal and phonemes is symmetrical about the current item. The contextual cue thus co-activates phonemes both before and after the current phoneme equally. By changing the strength of the association formed between the context signal and non-adjacent phonemes during encoding, the model is easily modified to exhibit persistency effects.

8.3.3 Separating Serial Control and Content

A further assumption made by the new inhibition model is motivated by the autosegmental theory of phonology (e.g. Kaye, 1989). Very briefly, autosegmental theory was motivated by the need for a more restricted theory of phonological processing than that of classical generative phonology (Chomsky & Halle, 1968). Autosegmental theory assumes that phonological information is organised in tiers (e.g. segmental, tonal, harmonic, syllabic) which are linked together by a level of timing units, called the *skeleton*. Processing occurs by the formation or deletion of association lines between each tier and the skeleton. Information from each tier cannot be directly linked, rather all associations are formed via the skeleton. Without going into too much detail about the properties and principles underlying autosegmental theory, the main point that is relevant to the model being developed here is that phonological structure is centred around a sequence of timing units which contain no phonetic information. The phonetic information is represented elsewhere among the tiers and associations between them and the skeleton. The separation of segmental information from its relative timing bears on the current model in the following way. It seems that the current oscillator based model is trying to do two things at once within the same mechanism. That is the model is not only performing the ordering of the phonemes, but also trying to remember which phonemes to assemble into the correct order as well. If an approach in a more autosegmental vein is taken, then it is justified to suppose that the segmental information receives activation from a source external to that which is relevant to the serial order mechanism. Thus the serial control of information can in a sense be partially separated from its content in a similar way that autosegmental theory specifies. This also has the advantage of restraining the model to one which is concerned mostly with the serial control of phonological information.

8.3.4 Method

A new model was implemented using a context signal with a step-size of three. This was associated with three-syllable phoneme sequences, of the form CVCVCV, to simplify the model. To compensate for the loss of performance which follows from using closer states of the context signal as cues for successive phonemes (i.e. a smaller step-size), synchrony within the context signal was increased to 75%. The three new extensions; inhibition, persistency and the separation of control from content, described above were also implemented, as follows.

The first extension, inhibition, was parsimoniously implemented by simply disadvantaging a phoneme by a small amount (typically 0.2) after it had been produced. Phonemes are output by choosing the phoneme whose similarity value with the current output is greatest. Thus after a phoneme was output during recall, the same phoneme was effectively inhibited by subtracting a small amount from its similarity value on subsequent time-steps. Persistency was the second extension and was implemented by decreasing the strength of the association (i.e. the learning rate parameter in the Hebbian learning) between the context signal and successive phonemes as the sequence of phonemes was learnt. This effectively ensured that phonemes produced before the current phoneme were more strongly represented than upcoming ones because phonemes early on in the sequence were encoded with a higher learning rate. The last extension, separating serial control of target segments from the activation of segmental content, was implemented by assuming that target phonemes received an advantage over non-target phonemes. This was actually implemented by partially suppressing all phonemes unless they were from the sequence being recalled. Specifically, phonemes that did not appear in the sequence had their similarity values reduced by a small amount during recall of the sequence. Phonemes in the sequence therefore had a small advantage of being output over phonemes not in the sequence. Thus the processing in the model was mainly focused

on the ordering of the target segments, since there was less chance of outputting phonemes not in the sequence (intrusions).

The procedure for learning and recall in the new model were the same as before, but now the model allowed phonemes to be inhibited, to have effects of persistency, and was more likely to recall only phonemes that appeared in the sequence, thus concentrating on the serial order of each sequence. For each simulation the new model was taught 5000 different phoneme sequences and overall performance and error classification were based on the level of performance achieved during the recall of each of the 5000 sequences.

8.3.5 Results

8.3.5.1 Overall Performance

The overall performance of the model is reasonable, recalling the right phoneme over 90% of the time for the initial phonemes, and over 80% of the time for the other phoneme positions. Figure 8.9 shows the change in performance with the position of every phoneme.

Performance is better at the beginning of the sequence and gradually decreases towards the end of the sequence. This is qualitatively different from the picture of overall performance obtained from the simple and synchrony models. This slight primacy effect can be explained by the persistency effect now in the new model. The model shows an effect of persistency because phonemes at the beginning of the sequence have stronger associations than later ones and hence are more likely to be correctly recalled than the ones with weaker associations.

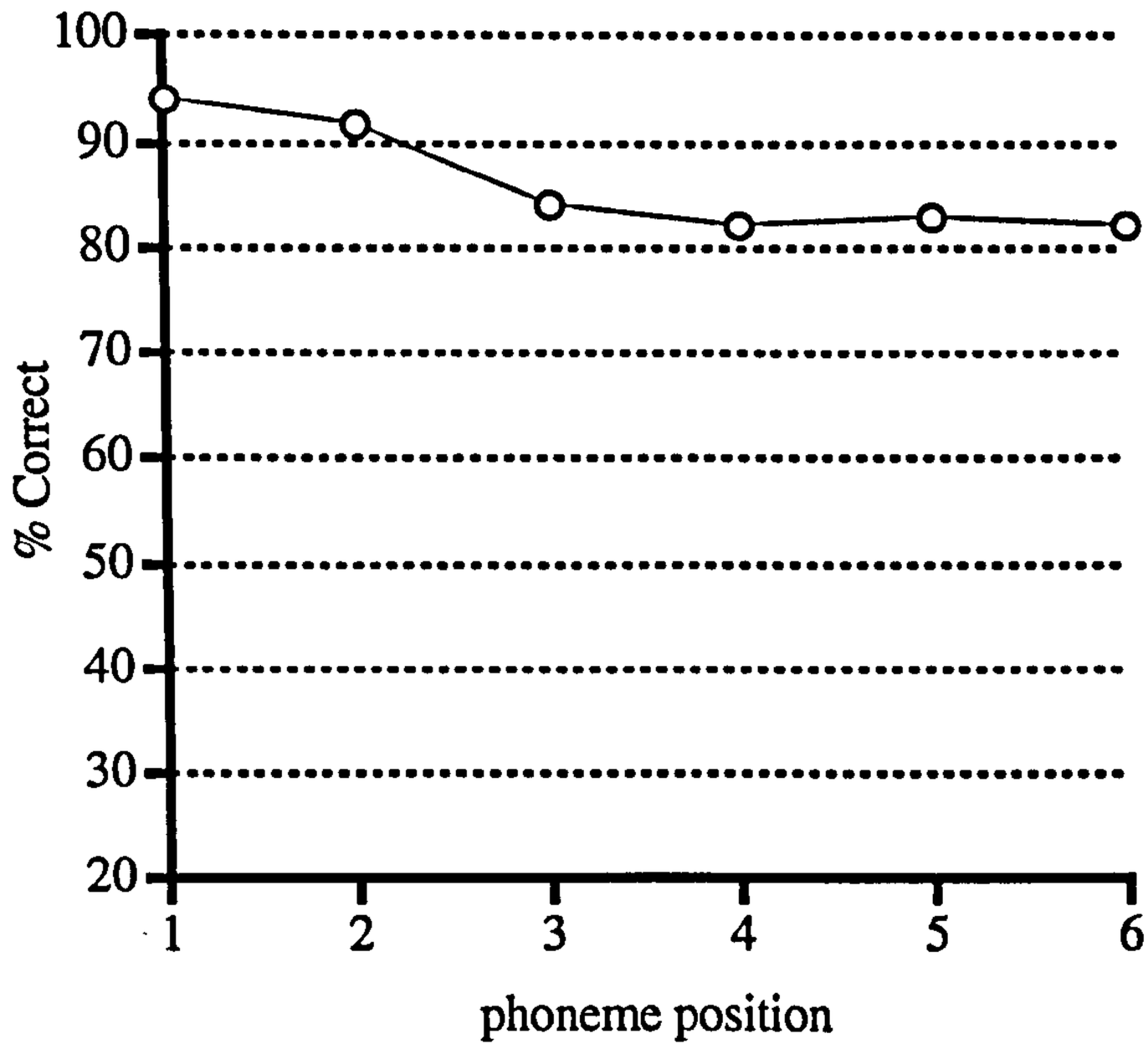

Figure 8.9. The overall performance of the inhibition model per phoneme position in a sequence (averaged over 5000 different random phoneme sequences of length six).

This effect on overall performance such that sequence-initial phonemes are recalled with greater accuracy is not found in the speech error data. Specifically, it contradicts the initialness effect (Dell et al., 1993) which finds that word-initial and syllable-initial phonemes are more prone to error than word-final or syllable-final phonemes.

8.3.5.2 Error Type Distribution

Figure 8.10 shows how the distribution of error types produced by the inhibition model compares with the human error data from Harley and MacAndrew's (1995) corpus. The model provides a good qualitative fit to the human speech error data and this can be accounted for by several factors. First, now that the additional burden of remembering which phonemes to assemble into the right order has been partially removed from the model, the distribution of movement errors (i.e. the errors produced by an imperfect serial ordering mechanism) becomes much clearer.

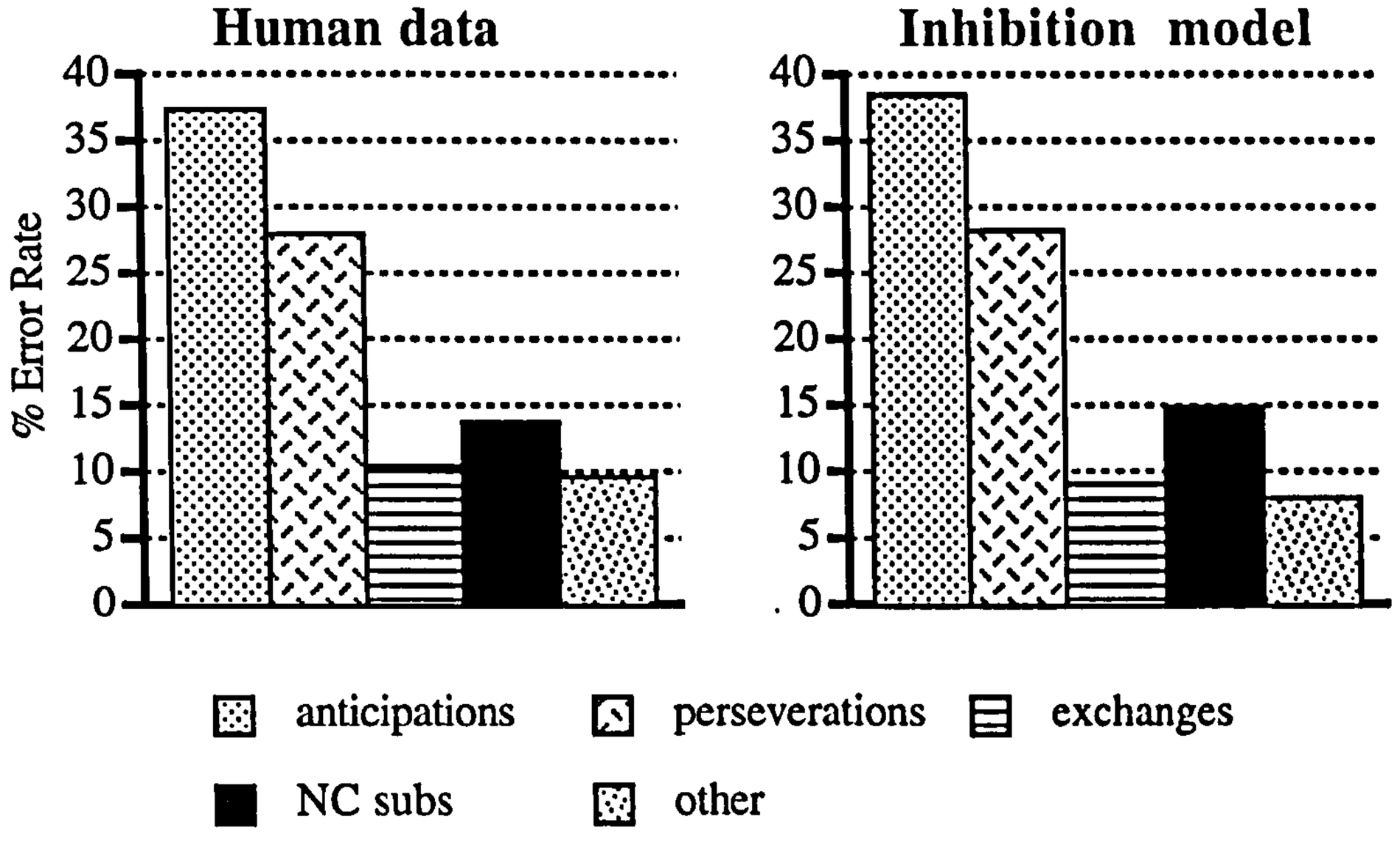

Figure 8.10. A comparison of the distribution of error types from the inhibition model with the human data. Each error category is shown as a percentage of total errors from the model (N=3765).

The number of non-contextual substitution errors has dropped to a level more in line with that which is observed in the human data. This means that the majority of the errors now produced will be movement errors. The proportion of errors that are exchanges is accounted for by the complementary partnership of inhibitory processing and persistency effects in the new model.

8.3.5.3 Syllable Constraint

Syllable position was generally preserved for all categories of movement errors. Anticipation errors respected syllabic position 99.9% of the time (N=1462), perseverations 100% (N=1073) and exchanges also 100% of the time (N=354).

8.3.5.4 Distance Constraint

The distance function of the exchange errors produced by the inhibition model is illustrated in Figure 8.11 (collapsed over consonant and vowel categories). Data

from Harley and MacAndrew's (1995) corpus has also been collapsed over consonant and vowel categories, and also over between-word and within-word exchange errors.

The pattern of separation from the model's errors is as predicted. Phonemes are more likely to exchange if they are temporally close together because the closer they are in time, the more co-activated they will be. This makes the task of choosing the correct phoneme more difficult. As the distance between phonemes increases, their co-activation becomes less apparent and the correct phoneme has fewer phonemes to compete with when it is due to be output. The gradient of the distance function is a little steeper than that of the human data, but still exhibits the correct qualitative result.

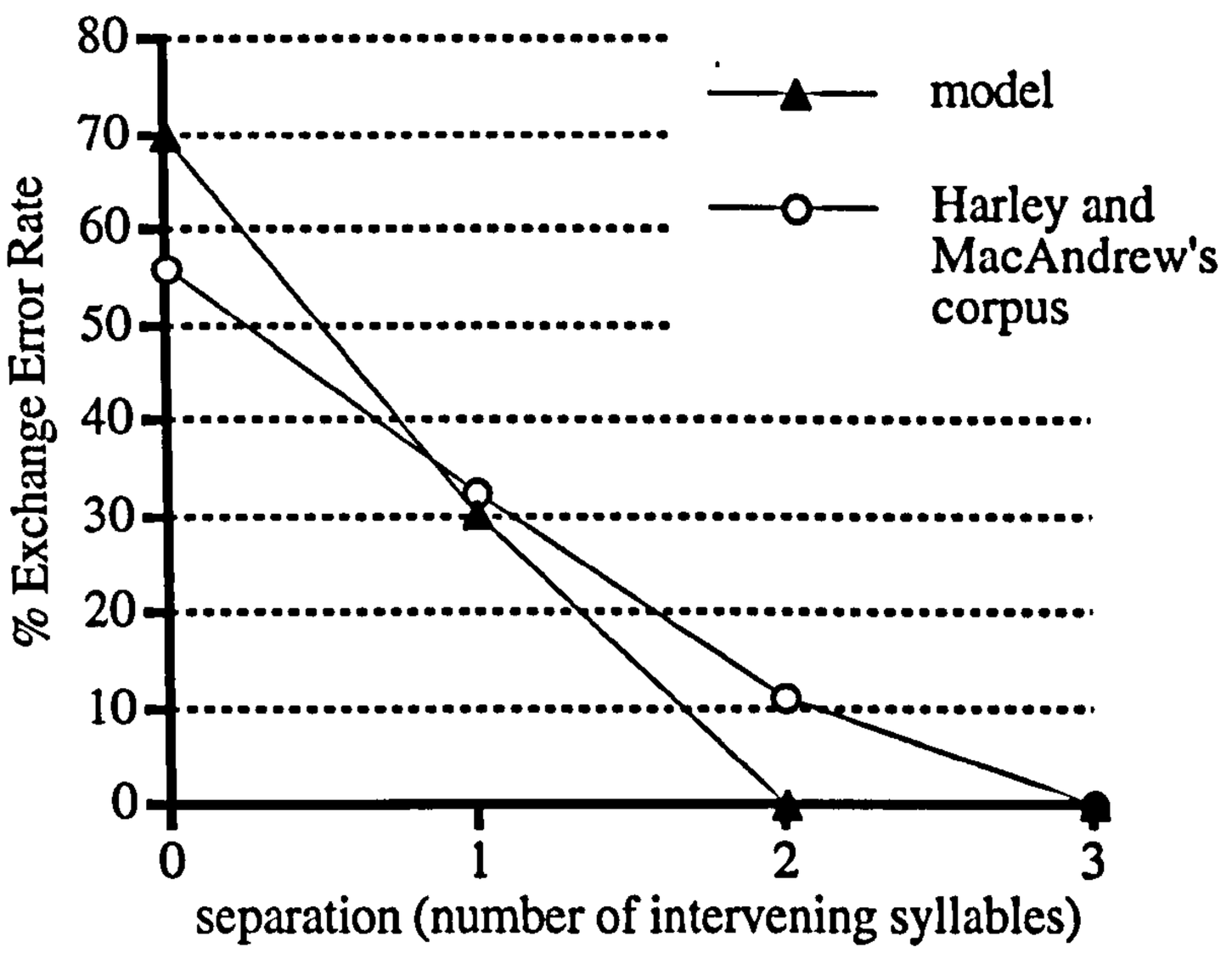

Figure 8.11. Proximity of exchange errors produced by the inhibition model (N=354) compared with data from Harley and MacAndrew's (1995) corpus. The exchange errors were collapsed over categories for between-word and within-word consonant and vowel errors with a separation of up to two syllables, for comparison with the model (N=224).

8.3.5.5 Phonetic Similarity Constraint of Exchanging Phonemes

The similarity of the phonemes involved in each exchange made by the model, as measured in terms of the number of feature dimensions on which they differed,

was calculated. The relative proportions of featurally different exchanging phonemes is shown in Figure 8.12, alongside the same gradient for the similarity of phonemes that exchange in the human data. Although the model does not provide a perfect fit to the data from Harley and MacAndrew's corpus, it exhibits the correct qualitative trend. That is the results indicate a monotonic decrease in the similarity of exchanging phonemes as a proportion of their frequency, so that exchanges are more likely to occur between phonemes that are close phonetic relatives. This is accounted for in the model by the way in which the phonemes are represented along different featural dimensions, in a distributed manner. The appropriate similarity between phonemes is thus captured by their featural representation. The similarity of phonemes facilitates exchanges because if two phonemes are very similar in nature and both have similar cues, then it is harder to distinguish them. If a phoneme is accidentally output too soon, then because of inhibitory processing it will also become suppressed immediately afterwards. This means that when the appropriate time comes to output the anticipated phoneme it will have a reduced activation. The target phoneme that was displaced by the anticipation will still be relatively highly activated, owing to the persistency effect. If it is also very similar to the anticipated phoneme then the chance of it being output (and hence of completing the exchange) is increased.

The relationship between the frequency of exchanges and their featural similarities, as is clear in Figure 8.12, appears as an emergent property of the model. The model produces a slightly better qualitative fit to the data from Meringer and Mayer's (1895) data on exchanges, as analysed by MacKay (1970), as can be seen from the graph to the right in Figure 8.12.

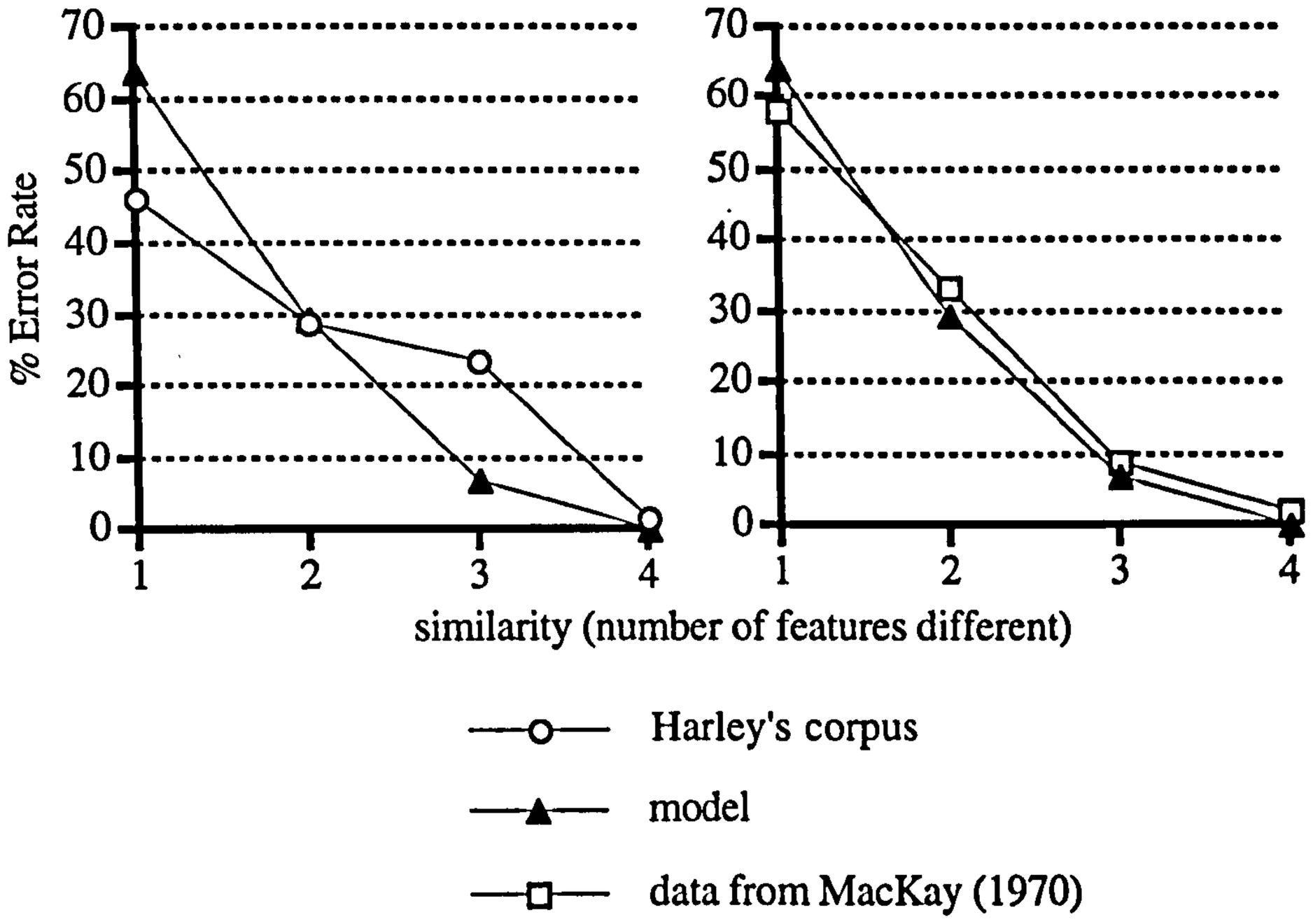

Figure 8.12. A comparison of the phonetic similarity of exchanging phonemes produced by the model (N=354) with those from Harley and MacAndrew's corpus (N=171) and data from Meringer and Meyer (1895) as analysed by MacKay (1970).

8.3.6 Discussion

The modifications to the model in this section show how the mechanisms of inhibitory processing and persistency bear on the nature of certain types of error. The introduction of inhibitory processing works by immediately suppressing a phoneme after is has been output, thus reducing its chances of being selected again unless it receives autonomous activation from other parts of the context signal. This reduces the chances of anticipatory errors being completed because the anticipated phoneme will immediately be suppressed. However a persistency effect means that phonemes that are displaced by another are more likely to be output (although in the wrong position) as soon as possible after they have been displaced, rather than later. When the two mechanisms operate harmoniously, the co-activation of phonemes in the sequence becomes closer to equality, therefore increasing the probability of an

exchange error occurring. This is what is seen in the inhibition model. The co-activation of phonemes in the model is primarily due to the dynamics of the context signal, although inhibition and persistency accentuate this fundamental property of the model. The model can now account for the exchange errors seen in Harley and MacAndrew's (1995) corpus, while still accounting for the proportions of errors that are anticipations and perseverations, as well.

8.4 Conclusion

At the beginning of this chapter, I set out to demonstrate how an oscillator based model could appropriately order speech sounds and account for some of the patterns of movement errors found in spontaneous speech errors, as formally described in chapter 2 and briefly summarised at the beginning of this chapter. To recap, these were the basic error distribution pattern, the syllable position constraint, the distance constraint for anticipations, perseverations and exchanges (the movement errors), and the phonetic similarity constraint between exchanging sounds. The fundamental property of the oscillatory based model that is central to this demonstration of serial control is its ability to co-represent phonemes in a sequence to a lesser or greater extent depending on the particular situation and sequence at hand.

In its most basic form, a simple model was described that made use of a new noisy context signal constructed from oscillators of different frequencies. The use of a noisy signal allowed greater discrimination between adjacent states of the context signal. The impact of a random element in the context signal was compensated for by increasing the dimensionality of the signal to remove any unwanted effects of noise and keep performance high. This was brought about by implementing a *virtual* context signal of the desired dimensionality. The majority of errors produced by this model were non-contextual substitution errors rather than movement errors, and although the distance constraint of the model's movement errors did provide a

reasonable qualitative fit to the data, the errors did not obey the syllable position constraint.

The synchrony model was motivated by the apparent need to introduce some sort of alignment for syllabic position into the model. Syllables were synchronised by including in the virtual context signal a proportion of context signals that repeated at a given frequency, thus providing a contextual cue for the alignment of each syllable. The temporal cue for syllables had the predicted effect of extending the similarity relationship of the context signal, in a hierarchical way, to sounds within each syllable so that syllable-initial sounds were more similar to other syllable-initial sounds than to syllable-final sounds. This was reflected in the results as a tendency for the syllable position constraint to be obeyed as the syllable alignment effect was increased. However, the problem of the large proportion of non-movement errors remained and lead to some additional mechanisms to be introduced into the model.

First, it was assumed that external activation for the form of the segments could be provided by another process, in a similar way perhaps to interactive activation models (e.g. Harley, 1993; Harley & MacAndrew, 1995). This was implemented by partially restricting recall to the phonemes in the particular sequence at hand, thus focusing on the serial ordering of phonemes rather than deciding which ones to order as well. Finally, to capture the change in phoneme activity as phonemes approach output and then are released from an activated state after output (cf. Houghton, 1992), inhibitory processing and persistency effects were included in the model. These mechanisms complement each other by encouraging a phoneme to remain active and be output as soon as possible unless it is output at the correct point. Support for these mechanisms as part of the speech production system was provided by a simulation in the last section of the chapter in which the new inhibition model displayed a more appropriate distribution of errors, providing a very good qualitative fit to the human data. The distance constraint obtained from the exchange errors showed the correct

pattern and a reasonable qualitative fit to the human data. Further support for the inhibition model came from the pattern of the featural similarity of exchanging phonemes, which showed a reasonable fit to the data from Harley and MacAndrew's corpus and an almost perfect fit (qualitatively) to the data from MacKay's (1970) analysis of Meringer and Mayer's (1895) corpus.

Chapter 9

9. Conclusion

The chapter begins with a summary of the motivations and work contained in this thesis. It describes how a model of the serial control of phonological information has been developed to meet the aims set out at its beginning. An interpretation of how the OSCAR model fits into current psycholinguistic theory and its implications follows. Finally, the limitations of the OSCAR model and possible modifications are discussed, providing suggestions for areas requiring further research.

9.1 Summary

The aim of this thesis was to further the understanding of the process which ensures that speech sounds are correctly ordered. This is the serial order problem: the process that controls the sequencing of phonemes as we speak. Such a process is apparent in normal speech and also from the existence of a class of speech errors known as sound movement errors, where sounds are anticipated (spoken too soon), perseverated (repeated again later), or exchanged (the sounds transpose). I have argued throughout that this process is temporally governed, that is it is restricted to processing sounds that are close together in time. This is in conflict with frame based accounts (e.g. Dell, 1986; Lapointe & Dell, 1979), serial buffer accounts (Shattuck-Hufnagel, 1979) and associative chaining theories (Wickelgren, 1969) of serial order.

The view of the serial order problem as a temporal process was motivated by speech error patterns mainly obtained from Harley and MacAndrew's (1995) corpus, and investigated by computational modelling techniques. The aim of the computational models was to simulate how the serial order problem may be solved. It was hoped that a solution could be achieved in terms of how temporal processing in a perfect system would operate and subsequently how and why the system would fail, producing a systematic pattern of sound movement errors.

9.1.1 The Need for Temporal Processes

Collections of speech errors reveal many constraints on the way in which errors can be produced and have provided motivation for general theories of production (Fromkin, 1971; Garrett, 1975, 1976) and for more specific theories relating to the serial order of phonemes (Lapointe & Dell, 1977; Shattuck-Hufnagel, 1979). In chapter 2, I described the key sound-based constraints on speech errors. These include the distance constraint, syllable position constraint, phonetic similarity constraint, the environment constraint and the phonotactic constraint. The distance constraint is particularly important because it is concerned with the temporal properties of speech which restricts movement errors to temporally close parts of speech. MacKay's (1970) analysis of Meringer and Mayer's (1895) sound exchange errors showed exchange errors were more likely to occur (above chance) the closer together they were. This clearly shows a tendency for errors to be temporally constrained, and therefore that a temporal element forms part of the serial order process. The distance constraint in movement errors is found not only in previous analyses (e.g. Garrett, 1975; MacKay, 1970), but also in a more recent analysis of Harley and MacAndrew's (1995) corpus, in chapter 2. An analysis of all anticipation, perseveration and exchange errors from Harley and MacAndrew's corpus in terms of the distance between the segments involved in each error revealed a qualitatively

similar pattern to that found in MacKay's (1970) analysis of Meringer and Mayer's (1895) corpus of sound exchanges. Specifically, anticipation and perseveration errors showed the same tendency to occur closer together in time than far apart, providing additional evidence for the hypothesis that all movement errors are temporally constrained. Other analyses of Harley and MacAndrew's (1995) sound movement errors revealed structural and phonetic constraints compatible with findings in other corpora, the two most relevant constraints to emerge being the syllable position constraint and the phonetic similarity constraint. This provided support for the idea that errors are also dependent on the phonetic similarity of nearby phonemes, and the syllabic position that they occupy.

Chapter 3 contained a detailed description of some models of speech production based on the patterns and constraints of speech error data. I focused on the sound sequencing aspects of these models, and categorised them into either frame-and-slot or buffer-based accounts on the one hand, or associative chaining accounts on the other. However, both types of model were argued to be inadequate in terms of their account of sequencing errors because they offer no plausible account for the distance constraint.

Associative chaining models rely on unidirectional associative links to produce sequential data. Such theories are dogged by problems concerning the formation of different sequences from the same elements because serial order is associated directly with each element. Thus associative chaining theories predict that errors are more likely to occur if the item preceding the error is repeated elsewhere in the same sequence. However the speech error data show that errors are equally likely to occur if the error segment is either preceded by or followed by a repeated element (Dell, 1986; MacKay, 1970). Empirical data has also shown that repeated elements influence the likelihood of error in non-adjacent positions as well as in adjacent positions (Dell, 1989). Chaining theories find these data problematic since the

sequencing information is contained in one link to the next item in sequence. Chaining theories also offer no account of structural constraints, such as the syllable position constraint.

Frame-based accounts on the other hand easily account for structural constraints such as the syllable position constraint, because they are effectively built into the representational assumptions made by the model. That is, frame-based models restrict filling of frames to specifically marked information. For example, a syllabic frame may contain slots for syllable onset, nucleus and coda information. Each slot would then only be connected to valid information of each type, such that the nucleus slot would only be connected to vowels, and so forth. Thus the syllable position constraint is automatically observed (e.g. Hartley & Houghton, 1994). Recently, Dell et al. (1993) showed how a neural network based frame-free account of speech production could account for data previously used as evidence for phonological frames. However, Dell et al.'s model could not account for movement errors.

In chapter 4, I reviewed the literature on computational modelling of temporal processing. These models were appealing because they directly addressed the issue of temporal processes during sequence production, and also because they provided a testable domain for theories of serial control mechanisms. They also addressed serial order as a temporal issue, rather than a spatial or frame-based one.

9.1.2 Discrete versus Simultaneous Temporal Representation

In chapter 5 I modelled some possible implementations of temporal processing, starting with Jordan's (1986b) recurrent network. Jordan's sequential network, like Dell et al.'s (1993) model does not assume phonological frames as part of the representational paradigm. Instead sequential order is achieved by serially activating each element in sequence at discrete time-steps. While Jordan's (1986b) model

provided a parsimonious paradigm for investigating temporal processing, it appeared to be limited by its computational complexity and the difficulty of the task at hand. For example, when the sequences shared repeating sub-sequences (as was the case in the Morse code problem), the model found it much harder to correctly recall the similar sequences. Modification of Jordan's (1986b) model with McLaren's (1993) APECS system in chapter 6 resulted in a perfect simulation of serial order with the REAPECS model of Morse code. The REAPECS model also performed well when applied to a speech production task. However, the REAPECS model did not behave as expected when lesioned, by systematically adding noise to various parts of the system. Instead of degrading gracefully, as is often characteristic of neural network models, the REAPECS model exhibited an all-or-nothing behaviour. On investigation, this was due to the way in which the serial order of each sequence was represented in the system. REAPECS had learnt sequences in much the same way that associative chaining theories represent sequential order, by single unidirectional links. Thus when sufficient random noise was introduced into the system, the links between all elements were equally likely to become active, resulting in a random sequence of elements, rather than a systematic pattern of errors. Specifically, movement errors were problematic for the model. So although a temporal processing system had been successfully modelled to produce serial order, it seemed to be little more than an implementation of associative chaining theory, and highly unlikely that the speech production system in humans could be of the same form. In order to account for the movement errors in human speech, it appeared that unless nearby phonemes were simultaneously represented at any particular time during production, the pattern of sound movement errors in speech would be very hard to explain.

Temporal processing during speech production is obviously necessary, but the simultaneous representation of temporally close phonemes is also apparently necessary to account for the movement of phonemes in errors. This is compatible

with an interactive activation, or competitive queuing, approach and incompatible with theories that postulate serial buffers (e.g. Shattuck-Hufnagel, 1979) or discrete representation of temporal processes (e.g. Jordan, 1986b; Dell et al., 1993). In chapter 7, I turned my attention to competitive queuing models which allow the simultaneous representation of sequential information, whose temporal order is determined by a dynamic control signal, that changes state with time (Houghton, 1990). Such models offer a more plausible paradigm for developing a temporally governed serial order mechanism because elements in a sequence are co-activated, depending on their relative temporal position. That is, elements closer together in time within a sequence are more strongly co-activated than elements far apart in the same sequence. Thus elements rise and fall in activation as their temporal position in the sequence passes and can be repeatedly output later in the same sequence without confusion because the serial order is determined primarily by the dynamic control signal rather than by associative links between each element. Thus phonemes are easily reactivated by the control signal later in the sequence if they are repeated. The notion of competitive queuing is attractive because only elements that are close together in time are co-activated, and therefore more likely to be involved in an error.

Simultaneous representation is apparent in speech error data, from the existence of sound movement errors, where sounds are systematically displaced by others from nearby in the utterance. The speech error data also suggest that phonological representations are stored across multiple dimensions, rather than at a single level, and all dimensions are interconnected and dependent to some extent. The results from the analyses of Harley and MacAndrew's (1995) corpus supported this, from the finding that movement errors are both temporally constrained and that other factors also play a role in the production of errors. A tiered representational view of phonology such as this is compatible with recent autosegmental theory (e.g. Berent & Perfetti, 1995; Ferreira, 1993; Kaye, 1989) and quite different from the classical

generative phonology of Chomsky and Halle (1968). Autosegmental theory advocates differently represented levels of phonological information all controlled and related by the skeleton, or timing tier. Processing occurs by the formation or deletion of association links between the levels, all via the skeleton.

Competitive queuing models, such as Houghton's (1990) model, allow associations to be formed between a control signal and a level containing segmental representations, similar to the timing tier and segmental level in autosegmental theory. However, Houghton's (1990) model lacks psychological motivation for the control signal and is limited by its low dimensionality. The phonetic similarity constraint is not captured by the model because the phonetic feature level of Houghton's model is not directly affected by the control signal. Furthermore, there is no plausible account of structural effects such as the syllable position constraint because there is nothing in Houghton's model to reflect the phonological structure found in human speech.

9.1.3 Oscillatory Based Temporal Control

There is however psychological motivation for timing processes controlled by oscillatory mechanisms from studies on time perception and motor action (e.g. Glass & MacKay, 1988; Treisman et al., 1994; Treisman et al., 1992). Experiments on time perception and motor actions, such as typing, have shown that when a task is accompanied by auditory clicks at certain critical rates, perturbations in time estimation occurs. This was assumed to be evidence of perturbation of an internal temporal oscillator. The resulting pattern of time estimation has also been used to gain estimates of the frequency of a temporal oscillator that governs time perception in humans. In chapters 7 and 8 I described in detail how a timing mechanism based on oscillatory devices could form the basis of a competitive queuing model, and further how such a model gave rise to the patterns of sound movement errors found in

human speech. This model was called OSCAR. An important mechanism in the OSCAR model described in chapter 7 was the clean 16-dimensional context signal, which operated such that states close together in time were very similar, and states far apart in time were proportionately less similar. The clean context signal always displayed a smooth, monotonic similarity function, and was useful in demonstrating the principles of OSCAR. A random element was introduced into the generation of the frequencies of the oscillators in the context signal to produce a less stable, or noisy similarity function. When a noisy context signal was associated to successive phonemes in a sequence, as described in chapter 8, the OSCAR model made more movement errors closer together in time than far apart, exhibiting a distance constraint like that found in the human error data. Further modifications to the model, motivated by synchronous processing in speech (Fowler, 1983), inhibitory processing (Dell & O'Seaghdha, 1993, Sevald & Dell, 1994), persistency effects (Page & Norris, 1995), and autosegmental phonology theory (Kaye, 1989), resulted in a qualitatively good demonstration of the distance constraint, the overall distribution of error types, the phonetic similarity constraint and the syllable position constraint.

9.2 Theoretical Issues and Implications

The computational models simulated in this thesis can be categorised into two distinct classes: recurrent network models and competitive queuing models. I have argued that the former are implementations of associative chaining to a lesser (e.g. Jordan, 1986b) or greater (e.g. the REAPECS model) extent and as such cannot account for the movement of phonemes in speech. The most promising model was based on the competitive queuing paradigm, which is compatible with a simultaneous representation of planned motor-actions. Most importantly, the OSCAR model described in chapters 7 and 8 shows how a separate timing representation (i.e. the context signal) controls the serial order of phonological information in speech. The

model raises issues for the representation and interaction of segmental information, and the range of articulatory planning.

The representation of consonants and vowels in the OSCAR model was similarly described, in that vowels were not represented independently from consonants. Both types of phonemes were represented along different articulatory dimensions, however. These articulatory dimensions (e.g. place of articulation, voicing etc.) may not be sufficient to capture the differences between the two types of phonemes. For example, there is evidence that consonants are articulated at a faster rate than vowels, and that the production of stressed vowels is cyclic and synchronous (Fowler, 1983), and even that consonants and vowels have different representational tiers (Berent & Perfetti, 1995). I have not addressed these issues in the OSCAR model, and have represented both vowels and consonants as similar phonemic types. Phonetic details are however represented separately from the timing and ordering information about each phonological sequence. This is compatible with an autosegmental approach to phonology rather than the classical generative phonology described in the seminal work of Chomsky and Halle (1968). More recent work from phonology advocates an autosegmental approach, where information is represented in a tiered system, controlled centrally by a timing tier called the skeleton. The model in chapter 8 describes how a timing tier could operate to control the serial order of phonemic sequences during production, where phonemes are associated with a timing level (the context signal) which is separate from their phonetic description. Below in a discussion of the limitations of the OSCAR model, the implications of similar representations of vowels and consonants are raised by the distribution of consonant and vowel errors produced by the model.

That the OSCAR model exhibits the distance constraint as observed in sound movement errors automatically implies that the speaking process at an articulatory level is temporally constrained. The active window of articulatory planning is

constrained to phonemes near together in time rather than at a distance. Furthermore, the co-activation of phonemes within the active window is proportional to their temporal relationship. Although a temporal constraint is obviously implied by the model, there is no evidence to suggest that the unit of say, syntactic planning in speech follows the same constraint. Other errors, such as whole word exchange errors are not constrained to the same temporal window as are sound errors, although for words to exchange implies that both words are simultaneously represented and therefore both currently planned. The model does show why the effective unit of planning at the articulatory level is much smaller than say, at the syntactic level, because the context signal is temporally constrained such that phonemes nearby in time will be co-activated more than distant phonemes.

The model's account of the phonetic similarity effect arises as an emergent property of the model. The model shows how phonetic similarity also influences the probability of phonemes exchanging, such that phonemes are more likely to exchange if they are also phonetically similar. When the features of nearby phonemes are co-activated, it is harder to identify the current phoneme if the other phonemes are very similar. The closer together the similar phonemes, the harder it becomes to discriminate between them, and hence the probability of them exchanging is increased. Most exchanges in the model occur between phonetically similar phonemes, as is also observed in the human data (MacKay, 1970).

The OSCAR model described in chapter 8 is clearly composed of separate mechanisms. These mechanisms are described separately as the synchrony model and the inhibition model. The synchrony mechanism accounts for the syllable position effect where error segments are constrained to occupy similar positions within a syllable. The complementary effects of inhibition, persistency, and separating order information from item information, account for the correct proportion of errors, the distance effect and phonetic similarity effect of the exchange errors. This modular

approach of the model has implications in the context of both psychology and neuropsychology in that it would be reasonable to look for a dissociation between the error effects accounted for by the two mechanisms. This would be particularly relevant in the area of aphasia, where the model would predict that a deficit in obeying syllabic structure could be dissociated from a tendency to produce certain error types in others. It is also possible that individual mechanisms in the model (e.g. the inhibition model, without any synchrony) could also account for empirical data from other areas of psychology (e.g. list learning in short-term memory). These are obvious possibilities for future research.

9.3 Limitations and Future Directions

The oscillator based model goes a long way in accounting for a range of speech error phenomena without recourse to explicitly defined phonological frames. The model has demonstrated how movement error data can be accounted for in a temporal system dynamically driven by a time varying contextual cue. However, there are certain properties of speech error data that are not accounted for by the model, and implementational aspects of the model that need discussion.

One undesirable result is that the model produces more vowel errors (as a proportion of consonant and vowel errors) than are observed in human speech. The difference between the proportions is small in the model, but in human speech the difference is much greater, with far fewer vowel than consonant errors reported. This could be an artefact of either the way sounds are represented in the model, or the form of the phoneme sequences the model was trained to recall. For example, a syllable may not contain more than one vowel but may contain several consonants. Thus it may be that the low number of vowel errors in human errors reflects the fact that more consonants are produced than vowels in normal speech and therefore one would expect a larger number of errors to be consonant errors. The model, on the other

hand, is trained to recall an equal number of consonants and vowels, which may be why the number of errors from each class is roughly equal. The smaller proportion of vowel errors in human speech may reflect a phonological frequency effect of the language. Alternatively, if, as Fowler (1984) suggests, vowels are produced in a slow, cyclic fashion and consonants are effectively superimposed onto the vowel signal, then it could also be that vowel forms are more robust than consonant articulations, or even represented differently. There is nothing in the model that reflects this hypothesis, which if it is correct, may be why vowels are as vulnerable to slip as consonants.

The model also has implementation limitations. As it is currently implemented the model's vocabulary is limited to one phoneme sequence. That is the model must be re-initialised with a different context signal and set of weights every time a different sequence is presented. A more desirable scenario would be a model that could represent many different sequences in the same system. This could be achieved by extending the model such that a larger set of oscillators existed, a different subset of which defined a different sequence. Thus the control signal for each different sequence would be distributed across a much larger set of oscillators. This would obviously be a worthwhile topic for future research. The learning algorithm (Hebbian learning) and the model architecture also place restrictions on the capacity of the model. The model consists of two layers of units, associated by Hebbian learning. If the problem space is not orthogonal (i.e. the input and output representations overlap) then the model may never find a solution to that problem, even though particular instances can be learnt. Thus each simulation of OSCAR can learn any particular word, but more than one word would cause too much interference for the model to correctly recall previously learnt words.

Relating to the vocabulary problem, the model is also limited to one phonological form of sequence. That is, there has been no variation from the

CV.CV.... sequence so far presented to the model. Obviously this places a large restriction on the errors the model can possibly make and a more realistic account should allow other phonological forms to be tested as well. The CV.CV... phonological form also simplifies the task of syllabification in the model, in that syllable boundaries are always guaranteed to fall after two phonemes. This also assumes that syllable durations are equal. Perhaps a more realistic approach would incorporate syllable synchrony in a temporally hierarchical form rather than a fixed durational form. This approach would allow different phonological forms of syllable to be represented, and is clearly also an area worthy of further research.

The issue of word-boundaries and the re-syllabification of words in the OSCAR model has not been addressed. For example, when two words such as "give it" are spoken consecutively, re-syllabification occurs such that the words are pronounced as "/gI.vIt./". The new syllable-boundaries do not correspond to the word-boundaries. Phoneme sequences in the model were assumed to be polysyllabic words whose serial order was controlled by the activation of the appropriate context signal. Thus different context signals were assumed for different words, but the syllable-boundary information was not directly associated with each word. Rather, syllable-boundaries in the model were formed by the inclusion of a synchronous context signal as part of the context signal for each word. Thus the *dynamics* of the context signal accounted for the positions of syllable-boundaries rather than its *content.* This is in contrast with models that store syllable information with each word (e.g. Dell, 1986), but consistent with theories where re-syllabification of words occurs on-line by firstly combining their metrical frames and then associating their segments to the new frame (e.g. Levelt, 1989; Levelt & Wheeldon, 1994). Thus the OSCAR model supports the view of syllabification as occurring on-line as part of the process of phonological encoding, the process of re-syllabification has not been explicitly addressed in the OSCAR

model. There is no representation of word-boundaries in the model, and the syllabification process assumes all syllables are of the form CV.CV.CV... etc.

Finally, the model does not address the issue of syntax. The function of the context signal is to control the serial order of phonological encoding and assumes lexical retrieval was successful. It does not address the process of lexicalisation nor any process concerned with the syntactic formulation of the utterance.

9.4 Concluding Comments

I have achieved my aim of providing an account of movement errors in speech production by demonstrating with the aid of computational models how temporal processing bears on the production of speech. Specifically, I have shown how parallel temporal processing in an oscillator based model can account for the movement of sounds in speech. This has been demonstrated to the extent that a similar prediction of the distance between error segments, their phonetic similarity and their position within the syllable is derived from the OSCAR model to the pattern of errors actually observed in speech error corpora. This has been demonstrated without recourse to an assumption of frame and slot structures, which I have argued are inadequate in their account of temporal processing effects. The OSCAR model on the other hand was able to account for these effects and other positional effects previously associated with frame based theories. The use of oscillator based models is still in its infancy, as is seen in the limitations of the OSCAR model. Oscillator dynamics, psychologically motivated by effects of physiological time-keeping mechanisms, seem an appropriate area for continued research, especially in speech production.

Appendix I: Morse Code

A	.-
B	-...
C	-.-.
D	-..
E	.
F	..-.
G	--.
H	....
I	..
J	.---
K	-.-
L	.-..
M	--
N	-.
O	---
P	.--.
Q	--.-
R	.-.
S	...
T	-
U	..-
V	...-
W	.--
X	-..-
Y	-.--
Z	--..

Appendix II: A Semantic description of words used in the REAPECS model

word	description (in semantic features)
beak	animate, 0-legs, small, component, hard, external, living
beet	plant, 0-legs, medium, edible, round, hard, living
bell	man-made, 0-legs, big, domestic, has-components, round, instrument, hard, metal
bolt	man-made, 0-legs, small, domestic, has-components, hard, metal
boot	man-made, 0-legs, big, domestic, hard, external, clothing
boss	animate, 2-legs, big, mobile, domestic, omnivore, male, female, old, young, human,living
cage	man-made, 0-legs, big, hard, metal, external
cake	man-made, 0-legs, medium, domestic, edible, round, soft
calf	animate, 4-legs, medium, mobile, domestic, edible, herbivore, male, female, young, living
cape	man-made, 0-legs, medium, domestic, component, soft, external, clothing
cart	man-made, big, mobile, domestic, has-components, hard, wood
cave	0-legs, big, old, young
chop	0-legs, medium, domestic, edible, sof
coin	man-made, 0-legs, small, domestic, round, old, young, hard, metal
coke	man-made, 0-legs, domestic, edible, soft, liquid
cone	man-made, 0-legs, medium, domestic, round, hard, soft
cork	plant, 0-legs, medium, domestic, round, component, soft, wood
crow	animate, flies, 2-legs, medium, mobile, omnivore, male, female, old, living
curb	man-made, 0-legs, medium, component, hard
dart	man-made, flies, 0-legs, small, domestic, has-components, hard, metal
deer	animate, 4-legs, big, mobile, edible, herbivore, male, old, living
dime	man-made, 0-legs, small, domestic, round, young, hard, metal
disc	man-made, 0-legs, medium, domestic, round, component, hard, soft, metal, wood
dock	man-made, 0-legs, big, has-components, aqua
doll	man-made, 2-legs, medium, domestic, male, female, old, young, hard
drum	man-made, 0-legs, medium, domestic, round, instrument, hard
duck	animate, flies, 2-legs, medium, mobile, aqua, edible, herbivore, male, female, old, living
fang	animate, 0-legs, small, old, young, component, hard, internal, living
flag	man-made, flies, 0-legs, medium, domestic, soft
flea	animate, 4-legs, small, mobile, domestic, omnivore, male, female, old, young, living
fork	man-made, 0-legs, medium, domestic, instrument, hard, metal
frog	animate, 4-legs, small, mobile, aqua, edible, herbivore, male, female, old, living
goat	animate, 4-legs, big, mobile, domestic, edible, herbivore, male, female, old, living
gown	man-made, 0-legs, medium, soft, external, clothing
hare	animate, 4-legs, medium, mobile, edible, herbivore, male, female, old, living
hawk	animate, flies, 2-legs, medium, mobile, carnivore, male, female, old, living

heel animate, 0-legs, medium, young, component, human, living
jeep man-made, 0-legs, big, mobile, domestic, has-components, instrument, hard
kilt man-made, 0-legs, medium, domestic, soft, external, clothing
kiss soft, human, living
kite man-made, flies, 0-legs, medium, mobile, domestic, has-components
knob man-made, 0-legs, medium, domestic, round, component, hard, metal, wood
lamb animate, 4-legs, small, mobile, domestic, edible, herbivore, male, female,young, living
lamp man-made, 0-legs, medium, domestic, has-components, instrument, hard, glass
lawn plant, 0-legs, big, domestic, has-components, soft, living
leaf plant, 0-legs, small, component, soft, living
lens man-made, 0-legs, medium, domestic, round, instrument, component, hard, glass
limb animate, medium, component, soft, external, living
lung animate, 0-legs, medium, component, soft, internal, human, living
mink animate, 4-legs, medium, mobile, omnivore, male, female, old, soft, living
mole animate, 4-legs, medium, mobile, herbivore, male, female, old, soft, living
moth animate, flies, 4-legs, small, herbivore, male, female, old, living
mule animate, 4-legs, big, mobile, domestic, herbivore, male, old, living
nail man-made, 0-legs, small, domestic, component, hard, metal, internal
nest man-made, 0-legs, medium, round, wood
pear plant, 0-legs, medium, domestic, edible, round, soft, living
pill man-made, 0-legs, small, domestic, edible, round, hard
pine plant, 0-legs, big, metal, liquid
pipe man-made, 0-legs, medium, domestic, instrument, hard, wood
plum plant, 0-legs, small, domestic, edible, round, soft, living
pole man-made, 0-legs, big, domestic, round, hard, metal, wood
pump man-made, 0-legs, medium, domestic, has-components, instrument, hard, metal
rake man-made, 0-legs, medium, domestic, instrument, hard, metal, wood
rope man-made, 0-legs, medium, domestic, round, soft
sail man-made, flies, 0-legs, big, soft
scab 0-legs, small, component, hard, external, human
scar 0-legs, medium, small, component, soft, external, human, living
seal animate, 0-legs, medium, mobile, aqua, carnivore, male, female, old, living
shed man-made, 0-legs, big, domestic, has-components, hard, wood
shoe man-made, 0-legs, medium, domestic, hard, external, clothing
sock man-made, 0-legs, small, domestic, soft, external, clothing
soup man-made, 0-legs, medium, domestic, aqua, edible, soft, liquid
stew man-made, 0-legs, medium, domestic, has-components, aqua, edible, soft
tank man-made, 0-legs, big, domestic, hard, metal
tent man-made, 0-legs, big, domestic, has-components
toad animate, 4-legs, small, mobile, aqua, herbivore, male, female, old, living
tomb man-made, 0-legs, big, has-components, old, hard
tray man-made, 0-legs, medium, domestic, hard, metal, wood
twig plant, 0-legs, small, component, hard, wood, living
vase man-made, 0-legs, medium, domestic, hard, glass
vest man-made, 0-legs, medium, domestic, soft, clothing
weed plant, 0-legs, small, living
whip man-made, medium, domestic, hard
wing animate, flies, 0-legs, medium, component, living
wolf animate, 4-legs, big, omnivore, male, female, old, living
wool domestic, soft
worm animate, 0-legs, small, living, mobile, old, young, soft

Appendix III: A Guide to Pronunciation Symbols

The symbols in the left hand column represent the sounds highlighted in bold in the words in the right hand column, based on the International Phonetic Alphabet. Hieroglyphic symbols have been replaced with letters from a QWERTY keyboard.

Consonants

b	**b**in
d	**d**in
f	**f**in
h	**h**at
k	**c**at
l	**l**ip
m	**m**at
n	**n**ot
p	**p**at
r	**r**at
s	**s**at
t	**t**in
v	**v**ine
w	**w**et
g	**g**ot
N	sing
T	**th**in
D	**th**en
S	**sh**ip
Z	vi**s**ion
j	**y**et
C	**ch**ip
J	**j**am

Vowels

i	**tree**
I	**sit**
e	**set**
@	**sat**
^	**sun**
a	**pass**
o	**dog**
c	**cord**
U	**put**
u	**do**
3	**bird**
eI	**late**
aI	**time**
cI	**boy**
3U	**toe**
aU	**cow**
I3	**dear**
e3	**care**
U3	**cure**

Appendix IV: Distinctive Features of English Phonemes

Phoneme pronunciation can be found in Appendix III. An explanation of the phonetic feature names in the table are given in the legend on the following page.

phoneme	phonetic feature name														
	cns	son	syl	hi	lo	bck	rnd	int	stri	nas	lat	vce	tns	cor	ant
b	+	-	-	-	-	-	-	+	-	-	-	+	*	-	+
d	+	-	-	-	-	-	-	+	-	-	-	+	*	+	+
f	+	-	-	-	-	-	-	-	+	-	-	-	*	-	+
h	-	-	-	-	+	-	-	-	-	-	-	-	*	-	-
k	+	-	-	+	-	+	-	+	-	-	-	-	*	-	-
l	+	+	-	-	-	-	-	-	-	-	+	+	*	+	+
m	+	+	-	-	-	-	-	+	-	+	-	+	*	-	+
n	+	+	-	-	-	-	-	+	-	+	-	+	*	+	+
p	+	-	-	-	-	-	-	+	-	-	-	-	*	-	+
r	+	+	-	-	-	-	-	-	-	-	-	+	*	+	-
s	+	-	-	-	-	-	-	-	+	-	-	-	*	+	+
t	+	-	-	-	-	-	-	+	-	-	-	-	*	+	+
v	+	-	-	-	-	-	-	-	+	-	-	+	*	-	+
w	-	+	-	+	-	+	+	-	-	-	-	+	*	-	-
z	+	-	-	-	-	-	-	-	+	-	-	+	*	+	+
g	+	-	-	+	-	+	-	+	-	-	-	+	*	-	-
N	+	+	-	+	-	+	-	+	-	+	-	+	*	-	-
T	+	-	-	-	-	-	-	-	-	-	-	-	*	+	+
D	+	-	-	-	-	-	-	-	-	-	-	+	*	+	+
S	+	-	-	+	-	-	-	-	+	-	-	-	*	+	-
Z	+	-	-	+	-	-	-	-	+	-	-	+	*	+	-
j	-	+	-	+	-	-	-	-	-	-	-	+	*	-	-
C	+	-	-	+	-	-	-	+	+	-	-	-	*	+	-
J	+	-	-	+	-	-	-	+	+	-	-	+	*	+	-
i	-	+	+	+	-	-	-	-	*	-	*	*	+	*	-
I	-	+	+	+	-	-	-	-	*	-	*	*	-	*	-
e	-	+	+	-	-	-	-	-	*	-	*	*	-	*	-
@	-	+	+	-	+	-	-	-	*	-	*	*	-	*	-
^	-	+	+	-	-	+	-	-	*	-	*	*	-	*	-
a	-	+	+	-	+	+	-	-	*	-	*	*	-	*	-
o	-	+	+	-	+	+	+	-	*	-	*	*	-	*	-
c	-	+	+	-	-	+	+	-	*	-	*	*	-	*	-
U	-	+	+	+	-	+	+	-	*	-	*	*	-	*	-
u	-	+	+	+	-	+	+	-	*	-	*	*	+	*	-
3	-	+	+	-	-	-	-	-	*	-	*	*	-	*	-

Legend

abbreviated feature name	feature name	definition
cons	consonantal	produced with contact between articulator & articulation
son	sonorant	vocal tract shaped so air flows unimpeded through nasal or oral cavity
syl	syllabic	segments with greatest prominence within a syllable
hi	high	tongue body above a neutral position
lo	low	tongue body below a neutral position
bck	back	retraction of tongue from neutral position
fnt	front	no retraction of the tongue
rnd	rounded	rounding of the lips
int	interrupted	a complete blockage of the airstream during part of an articulation
stri	strident	vocal tract shaped so air only flows through a narrow gap in the centre
nas	nasal	some or all of the air is expelled through the nose
lat	lateral	airstream is diverted laterally around the tongue
vce	voiced	periodic vibrations of the vocal cords
tns	tense	relatively high tension in the oral cavity muscles
cor	coronal	front or apex of tongue is raised to form a total or partial obstruction
ant	anterior	sounds made at or in front of the alveolar ridge

Appendix V: The Phonetic Form of Words Used in the REAPECS model

word	phonetic form (based on phonetic symbols in Appendix III)
beak	bik
beet	bit
bell	bel
bolt	b3Ult
boot	but
boss	bos
cage	keIJ
cake	keIk
calf	kaf
cape	keIp
cart	kat
cave	keIv
chop	Cop
coin	koIn
coke	k3Uk
cone	k3Un
cork	kck
crow	kr3U
curb	k3b
dart	dat
deer	dI3
dime	daIm
disc	dIsk
dock	dok
doll	dol
drum	drm
duck	d^k
fang	f@N
flag	fl@g
flea	fli
fork	fck
frog	frog
goat	g3Ut
gown	gaUn
hare	he3
hawk	hck
heel	hil
jeep	Jip
kilt	kIlt
kiss	kIs
kite	kaIt
knob	nob
lamb	l@m
lamp	l@mp

lawn	lcn
leaf	lif
lens	lenz
limb	lIm
lung	l^N
mink	mINk
mole	m3Ul
moth	moT
mule	mjul
nail	neIl
nest	nest
pear	pe3
pill	pIl
pine	paIn
pipe	paIp
plum	pl^m
pole	p3Ul
pump	p^mp
rake	reIk
rope	r3Up
sail	seIl
scab	sk@b
scar	ska
seal	sil
shed	Sed
shoe	Su
sock	sok
soup	sup
stew	stju
tank	t@Nk
tent	tent
toad	t3Ud
tomb	tum
tray	treI
twig	twIg
vase	vaz
vest	vest
weed	wid
whip	wIp
wing	wIN
wolf	wUlf
wool	wUl
worm	w3m

References

Baars, B.J., Motley, M.T., & MacKay, D.G. (1975). Output editing for lexical status in artificially elicited slips of the tongue. *Journal of Verbal Learning and Verbal Behavior, 14,* 382-391.

Bairaktaris, D. (1994). The problem of temporal order in connectionist networks and its implications for short-term memory modelling. In M. Oaksford & G.D.A. Brown (Eds.), *Neurodynamics and Psychology.* London: Academic Press.

Berent, I. & Perfetti, C.A. (1995). A rose is a REEZ: The two-cycles model of phonology assembly in reading English. *Psychological Review, 102,* 146-184.

Beaufays, F. & Wan, E.A. (1994). Relating real-time back-propagation and BPTT: An application of flow-graph interreciprocity. *Neural Computation, 6,* 296-306.

Billon, M., & Semjen, A. (1995). The timing effects of accent production in synchronization and continuation tasks performed by musicians and nonmusicians. *Psychological Research, 58,* 206-217.

Bock, K. (1987). Exploring levels of processing in sentence production. In G. Kempen (Ed.), *Natural Language Generation.* Dordrecht, Netherlands: Kluwer Academic Publishers.

Boomer, D.S., & Laver, J.D.M. (1968). Slips of the tongue. *British Journal of Disorders of Communication, 3,* 2-12.

Broeke, Van Den M.P.R., & Goldstein, L. (1980). Consonant features in speech errors. In V.A. Fromkin (Ed.), *Errors in Linguistic Performance: Slips of the Tongue, Ear, Pen and Hand.* London: Academic Press.

Browman, C.P. (1980). Perceptual processing: evidence from slips of the ear. In V. A. Fromkin (Ed.), *Errors in Linguistic Performance: Slips of the Tongue, Ear, Pen and Hand.* London: Academic Press.

Brown, R., & McNeill, D. (1966). The tip of the tongue phenomena. *Journal of Verbal Learning and Verbal Behavior, 5,* 325-337.

Brown, G.D.A., Preece, T., & Hulme, C. (1996). An oscillator-based model of memory for serial order. Unpublished manuscript.

Burgess, N. , & Hitch, G.J. (1992). Toward a network model of the articulatory loop. *Journal of Memory and Language, 31,* 429-460.

Butterworth, B. (1982). Speech errors: old data in search of new theories. In A. Cutler (Ed.), *Slips of the tongue and language production.* Amsterdam: Mouton.

Butterworth, B. (1989). Lexical access in speech production. In W. Marslen-Wilson (Ed.), *Lexical representation and process.* Cambridge, MA: MIT Press.

Chater, N. (1989). *Learning to respond to structure in time.* Unpublished manuscript.

Chater, N., & Conkey, P. (1994). Sequence processing with recurrent neural networks. In M. Oaksford & G. D. A. Brown (Eds.), *Neurodynamics and Psychology.* London: Academic Press.

Chomsky, N., & Halle, M. (1968). *The Sound Pattern of English.* New York: Harper and Row.

Cutler, A. (1982). The reliability of speech error data. In A. Cutler (Ed.), *Slips of the Tongue and Language Production.* Amsterdam: Mouton.

Das, S., Giles,C.L., & Sun, G.Z. (1992). Using Prior Knowledge in a NNPDA to Learn Context-free languages. In *Advances in Neural Information Processing Systems 5.* San Mateo, CA: Morgan Kaufmann.

Dell, G.S. (1984). Representation of serial order in speech: Evidence from the repeated phoneme effect in speech errors. *Journal of Experimental Psychology: Learning, Memory and Cognition, 10,* 222-233.

Dell, G.S. (1986). A spreading activation theory of retrieval in sentence production. *Psychological Review, 93,* 283-321.

Dell, G.S. (1989). The retieval of phonological forms in production: Tests of predictions from a connectionist model. In W. Marslen-Wilson (Ed.), *Lexical representation and process*. Cambridge, MA: MIT Press.

Dell, G.S., & O'Seaghdha, P. G. (1993). Inhibition in interactive activation models of linguistic selection and sequencing. In D. Dagenbach and T.H. Carr (Eds.), *Inhibitory Processes in Attention, Memory and Language*. San Diego, CA: Academic Press.

Dell, G.S., & Reich, P.A. (1981). Stages in sentence production: An analysis of speech error data. *Journal of Verbal Learning and Verbal Behavior, 20,* 611-629.

Dell, G.S., Burger, L.K., & Svec, W. (1995). *Language production and serial order: A functional analysis and a model.* Manuscript submitted for publication.

Dell, G.S., Juliano, C., & Govindjee, A. (1993). Structure and content in language production: A theory of frame constraints in phonological speech errors. *Cognitive Science, 3(2),* 128-138.

Ellis, A.W. (1980). Errors in speech and STM: The effects of phoneme similarity and syllabic position. *Journal of Verbal Learning and Verbal Behavior, 19,* 624-634.

Elman, J.L. (1990). Finding structure in time. *Cognitive Science, 14,* 179-211.

Elman, J.L. (1991). *Incremental learning, or the importance of starting small.* CRL Tech. Report 9101, Centre for Research in Language, University of California, San Diego, La Jolla, CA.

Fahlman, S.E. (1991). *The recurrent cascade correlation architecture.* Unpublished manuscript, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.

Fay, D., & Cutler, A. (1977). Malaproprisms and the structure of the mental lexicon. *Linguistic Inquiry, 8,* 505-520.

Ferber, R. (1990). Slip of the tongue or slip of the ear? On the perception and transcription of naturalistic slips of the tongue. *Journal of Psycholinguistic Research, 20,* 105-122.

Ferreira, F. (1993). Creation of prosody during sentence production. *Psychological Review, 100,* 233-253.

Fowler, C.A. (1979). "Perceptual centers" in speech production and perception. *Perception and Psychophysics, 25,* 375-388.

Fowler, C.A. (1983). Converging sources of evidence on spoken and perceived rhythms of speech: Cyclic prodution of vowels in monosyllabic stress feet. *Journal of Experimental Psychology: General, 112,* 386-412.

Fromkin, V.A. (1971). The non-anomalous nature of anomalous utterances. *Language, 51,* 27-52.

Garnham, A., Shillcock, R.C., Brown, G.D.A., Mill, A.I.D., & Cutler, A. (1982). Slips of the Tongue in the London-Lund corpus of spontaneuous conversation. In A. Cutler (Ed.), *Slips of the Tongue and Language Production.* Amsterdam: Mouton.

Garrett, M.F. (1975). The analysis of sentence production. In G. Bower (Ed.), *Psychology of learning and motivation, 9,* 137-177. New York: Academic Press.

Garrett, M.F. (1976). Syntactic process in sentence production. In R. J. Wales and E. Walker (Eds.), *New Approaches to Language Mechanisms.* Amsterdam: North Holland Publishing Complany.

Gilhooly, K.J., & Logie, R.H. (1980). Age of acquisition, imagery, concreteness, familiarity and ambiguity measures for 1944 words. *Behavior Research Methods and Instrumentation,12,* 395-427.

Gimson, A.C. (1980). *An Introduction to the Pronunciation of English.* London: Edward Arnold.

Glass, L., & Mackey, M.C. (1988). *From clocks to chaos: The rhythms of life.* Princeton, NJ: Princeton University Press.

Goldman-Eisler, F. (1958). Speech production and the predictability of words in context. *Quarterly Journal of Experimental Psychology, 10,* 96-106.

Goldman-Eisler, F. (1968). *Psycholinguistics: Experiments in spontaneous speech.* London: Academic Press.

Harley, T.A. (1984). A critique of top-down independent levels models of speech production: Evidence from non-plan-internal speech errors. *Cognitive Science, 8,* 191-219.

Harley, T. A. (1990). Paragrammatisms: Syntactic disturbance or failure of control ? *Cognition, 34,* 85-91.

Harley, T. A. (1993a). Connectionist approaches to language disorders. *Aphasiology, 7,* 221-249.

Harley, T.A. (1993b). Phonological activation of semantic competitors during lexical access in speech production. *Language and Cognitive Processes, 8(3),* 291-309.

Harley, T.A., & MacAndrew, S.B.G. (1992). Modelling paraphasias in normal and aphasic speech. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society,* J. K. Kruschke (Ed.), 378-383. Hillsdale, NJ: Earlbaum.

Harley, T.A., & MacAndrew, S.B.G. (1995). Interactive models of lexicalisation: Some constraints from speech error, picture naming, and neuropsychological data. In J. Levy, D. Bairaktaris, J. Bullinaria, & D. Cairns (Eds.), *Connectionist models of memory and language*. London: UCL Press.

Hartley, T. , & Houghton, G. (1994). *A linguistically constrained model of short-term memory for non-words.* Manuscript submitted for publication.

Hinton, G.E., & Sejnowski, T.E. (1983). Optimal perceptual inference. *Proceedings of the Institute of Electronic and Electrical Engineers Computer Society Conference on Computer Vision and Pattern Recognition.* Washington, DC, IEEE, 448-53.

Hinton, G.E., & Sejnowski, T.E. (1986). Learning and relearning in Boltzmann machines. In D. E. Rumelhart and J.L. McClelland (Eds.), *Parallel distributed processing, I:Foundations.* Cambridge, MA: MIT Press.

Hinton, G.E., & Shallice, T. (1991). Lesioning an attractor network: Investigations of acquired dyslexia. *Psychological Review, 98,* 74-95.

Hopfield, J.J. (1982). Nerual networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA, 79,* 2554-2558.

Houghton, G. (1990). The problem of serial order: A neural network model of sequence learning and recall. In R. Dale, C. Mellish & M. Zock (Eds.), *Current Research in Natural Language Generation.* London: Academic Press.

Houghton, G. (1992). Inhibitory control of neurodynamics: Opponent mechanisms in sequencing and selective attention. In M. Oaksford and G.D.A. Brown (Eds.), *Neurodynamics and Psychology.* London: Academic Press.

Houghton, G., & Hartley, T. (1995). *Parallel models of serial behaviour: Lashley revisited.* Unpublished manuscript.

Houghton, G. & Tipper, S.P. (1994). A model of inhibitory mechanisms in selective attention. In D. Dagenbach and T.H. Carr (Eds.), *Inhibitory Processes in Attention, Memory and Language.* San Diego, LA: Academic Press.

Houghton, G., Glasspool, D.W., & Shallice, T. (1994). Spelling and serial recall: Insights from a competitive queuing model. In G.D.A. Brown and N.C. Ellis (Eds.), *Handbook of Spelling: Theory, Process and Intervention.* New York: John Wiley and Sons Ltd.

Jordan, M.I. (1986a). *Serial order: A parallel, distributed approach.* Technical Report No. 8604, University of California, La Jolla, San Diego.

Jordan, M.I. (1986b). Attractor dynamics and parallelism in a connectionist sequential machine. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Erlbaum.

Jordan, M.I. (1989a). Generic constraints on underspecified target trajectories. *Proceedings of the International Joint Conference on Neural Networks.*

Jordan, M.I. (1989b). Serial order: A parallel, distributed processing approach. In J. L. Elman and D. E. Rumelhart (Eds.), *Advances in Connectionist Theory: Speech.* Hillsdale, NJ: Erlbaum.

Jordan, M.I., & Rumelhart, D.E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive Science, 16,* 307-354.

Kaye, J. (1989). *Phonology: A cognitive view.* Hillsdale, NJ: Earlbaum.

Kohn, S.E., & Smith, K.L. (1994). Distinctions between two phonological outut deficits. *Applied Psycholinguistics, 15,* 75-95.

Kohn, S.E., & Smith, K.L. (1995). Serial effects of phonemic planning during word production. *Aphasiology, 9,* 209-222.

Kucera, H., & Francis, W.N. (1967). *Computational analysis of present-day American English.* Providence, NJ: Brown University Press.

Lapointe, S.G., & Dell, G.S. (1989). A synthesis of some recent work in sentence production. In G.N. Carlson & M.K. Tananhaus (Eds.), *Linguistic Structure in Language Processing.* Dordrecht, Netherlands: Kluwer Academic Publishers.

Lashley, K.S. (1951). The problem of serial order in behaviour. In L. A. Jefress (Ed.), *Cerebral mechanisms in behaviour.* New York: Wiley.

Lass, R. (1984). *Phonology: An Introduction to Basic Concepts.* Bath, England: Pitman Press.

Levelt, W.J.M. (1989) *Speaking: From intention to articulation.* Cambridge, MA: MIT Press.

Levelt, W.J.M., Schreifers, H., Vorberg, D., Meyer, A.S., Pechmann, T., & Havinga, J. (1991). The time course of lexical access in speech production: A study of picture naming. *Psychological Review, 98,* 122-142.

Levelt, W.J.M., & Wheeldon, L. (1994). Do speakers have access to a mental syllablary? *Cognition, 50,* 239-269.

MacKay, D. (1970). Spoonerisms: The structure of errors in the serial order of speech. *Neuropsychologia, 8,* 323-350.

McClelland, J.L., & Rumelhart, D.E. (1981). An interactive activation model of context effects in letter perception: Part I. An account of basic findings. *Psychological Review, 88,* 375-407.

McCloskey, M., & Cohen, N.J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In G.H. Bower (Ed.), *The psychology of learning and motivation.* New York: Academic Press.

McLaren, I.P.L. (1993). APECS: a solution to the sequential learning problem. *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society,* 717-722.

Meringer, R., & Mayer, K. (1895). *Versprechen und Verlesen: Eine Psychologische-Linguistische Studie.* Stuttgart: Gössen.

Meyer, A.S. (1990). The time course of phonological encoding in language production: The encoding of successive syllables of a word. *Journal of Memory and Language, 29,* 524-545.

Meyer, A.S. (1991). The time course of phonological encoding in language production: Phonological encoding inside a syllable. *Journal of Memory and Language, 30,* 69-89.

Meyer, A.S. (1992). Investigation of phonological encoding through speech error analyses: Achievements, limitations and alternatives. *Cognition, 42,* 181-211.

Meyer, A.S. (1994). Timing in sentence production. *Journal of Memory and Language, 33,* 471-492.

Millar, G.A., & Nicely, P.E. (1955). An analysis of perceptual confusions among some english consonants. *Journal of the Acoustical Society of America, 27,* 338-352.

Page, M.P.A., & Norris, D. (1995). *The primacy model: A new model of immediate serial recall.* Manuscript submitted for publication.

Paivo. Unpublished Manuscript, as cited in Quinlan, (1992).

Petrie, H. (1987). The psycholinguistics of speaking. In J. Lyons, R. Coates, M. Deuchar, & G. Gazdar (Eds.), *New Horizons in Linguistics, 2,* 336-367.

Plaut, D.C. & Shallice, T. (1993). Deep dyslexia: A case study of connectionist neuropsychology. *Cognitive Neuropsychology, 10,* 372-500.

Plaut, D.C., & McClelland, J.L. (1993). Generalisation with componential attractors: Word and non-word reading in an attractor network. *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society, Boulder, Colorado.* W. Kintsh (Ed.). New York: Erlbaum.

Quinlan, P.T. (1992). *Oxford Psycholinguistic Database.* Oxford, England: Oxford University Press.

Rumelhart, D.E., & McClelland, J.L. (1986). *Parallel distributed processing, I:Foundations.* Cambridge, MA: MIT Press.

Rumelhart, D.E., & Norman, D.A. (1982). Simulating a skilled typist: A study of skilled motor performance. *Cognitive Science, 6,* 1-36.

Rumelhart, D.E., Hinton, G., & Williams, R.J. (1986). Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel distributed processing, I:Foundations.* Cambridge, MA: MIT Press.

Seidenberg, M.S., & McClelland, J.L. (1989). A distributed, developmental model of word recognition and naming. *Psychological Review, 96,* 523-568.

Sejnowski, T.J., & Rosenberg, C.R. (1986). Parallel networks that learn to pronounce English text. *Complex Systms, 1,* 145-168.

Servan-Schreiber, D., Cleermans, A., & McClelland, J.L. (1991). Graded state machines: The representation of temporal contingencies in simple recurrent networks. *Machine Learning, 7,* 161-193.

Sevald, C.A., & Dell, G.S. (1994). The sequential cueing effect in speech production. *Cognition, 53,* 91-127.

Shattuck-Hufnagel, S. (1979). Speech errors as evidence for a serial-ordering mechanism in sentence production. In W.E. Cooper & E.C.T. Walker (Eds.),

Sentence Processing: Psycholinguistic Studies Presented to Merrill Garrett. Hillsdale, NJ: Erlbaum.

Shattuck-Hufnagel, S., & Klatt, D.H. (1979). The limited use of distinctive features and markedness in speech production: Evidence from speech error data. *Journal of Verbal Learning and Verbal Behavior, 18,* 41-55.

Singh, S. (1976). *Distinctive Features, Theory & Validation.* Baltimore: University Park Press.

Sloat, C., Henderson-Taylor, S. & Hoard, J.E. (1978). *Introduction to Phonology.* Englewood Cliffs, NJ: Prentice-Hall.

Stemberger, J.P. (1982). The nature of segments in the lexicon: Evidence from speech errors. *Lingua, 56,* 235-259.

Stemberger, J.P. (1985). An interactive activation model of language production. In A. Ellis (Ed.), *Progress in the Psychology of Language* (Vol 1, pp.143-186). London: Erlbaum.

Stemberger, J.P. (1990). Wordshape errors in language production. *Cognition, 35,* 123-157.

Toglia, M.P., & Battig, W.R. (1978). *Handbook of Semantic Word Norms.* New York: Earlbaum.

Treisman, M., Cook, N., Naish, P.L.N., & MacCrone, J.K. (1994). The internal clock: Electroencephalographic evidence for oscillatory processes underlying time perception. *The Quarterly Journal of Experimental Psychology, 47A,* 241-289.

Treisman, M., Faulkner, A., & Naish, P.L.N. (1992). On the relation between time perception and the timing of motor action: Evidence for a temporal oscillator controlling the timing of movement. *The Quarterly Journal of Experimantal Psychology, 45A,* 235-263.

Wang, D., & Arbib, M.A. (1993). Timing and chunking in processing temporal order. *In IEEE transactions on systems, man, and cybernetics, 23,* 993-1009.

Wells, R. (1951). Predicting slips of the tongue. *Yale Scientific Magazine, December,* 9-12.

Wickelgren, W.A. (1965). Distinctive features and errors in short-term memory for English vowels. *Journal of the Acoustical Society of America, 38,* 583-588.

Wickelgren, W.A. (1966). Distinctive features and errors in short-term memory for english consonants. *Journal of the Acoustical Society of America, 39,* 388-398.

Wickelgren, W.A. (1969). Context-sensitive coding, associative memory, and serial order in (speech) behaviour. *Psychological Review, 76,* 1-15.

Williams, R.J., & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation, 1,* 270-280.