LINEAR MODELS OF LANGUAGE PROCESSING

by

Douglas P. Wilson
B.G.S., Simon Fraser University, 1979

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ARTS
(Special Arrangement)

SIMON FRASER UNIVERSITY
April 1984

APPROVAL

Name: Douglas P. Wilson

Degree: Master of Arts (Special Arrangement)

Title of Thesis: Linear Models of Language Processing

Examining Committee:

Chairman: Dr. M.W. Bowman

Dr. T.W. Calvert
Senior Supervisor

Dr. H. Weinberg
Psychology Department

Dr. R.E. Jennings
Philosophy Department

Dr. E.W. Roberts
Department of L.L.L.

Dr. M.P. Beddoes
External Examiner
Department of Electrical Engineering
University of British Columbia

Date Approved: April 5, 1984

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

"Linear Models of Language Processing"

Author: _____
(signature)

Douglas P. Wilson
(name)

December 6th/1985
(date)

ABSTRACT

The advantages of linear devices as communications channels suggest the need for a linear model of language processing. Linear devices have a unique pseudoinverse, so expressions in language should have a single most likely interpretation. The contrary view that human language is ambiguous should be reconsidered with a new understanding of the role of context.

Computer studies of linear models began with an application of the least-effort principle to feature extraction. Alphabetical symbols were treated as representing neighbourhoods in the output space of a linear device, with text being treated as an encoding of a continuous section of a trajectory through the space. The constraints this imposes on the relative positions of neighbourhoods provide a set of coordinate values for the letters of the alphabet similar to the distinctive features of generative phonology, but without reference to prior knowledge or linguistic intuitions.

Using coordinate values obtained in this way, text was translated into multivariate functions, filtered with various simple digital filters, and retranslated into text, in an attempt to mimic grammatical processes. In performing these operations it was noticed that certain sequences of letters would pass through the filtering operation unchanged, while others were changed only from an uncommon sequence to a more common one. This suggested the construction of a linear grammatical filter which would have no effect on acceptable

sequences of letters, while altering others.

Such a device was implemented using a projection onto the subspace containing the most common trajectories. The similarity of the resulting grammatical filter to Markov devices was noted, and a reformulation of the latter as filters described. The limitations of finite-state Markov sources can be overcome with a filtrative model. Other devices from automata theory can also be treated as filters, and some of these will be linear. It is argued that the distinction between linear and nonlinear devices is more relevant to linguistics than the distinctions between various classes of automata as originally proposed by Chomsky.

Psychological consequences of this view include a new appreciation of associationism. Possible models of the brain as a high-order linear filter are described and evaluated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

The material covered in this thesis bridges several disciplines, and seems hard to understand. In order to make it as approachable as possible, I have divided the thesis into three parts. Part One presents a theory of language and some experimental results. This part is made straightforward and concise by omitting difficult areas. Hard parts of the theory and more details of the experiments are given in Part Two, which includes short sections on a number of topics related to the earlier material. Part Three contains a summary and conclusions.

The theory of language presented in Part One is based on methods of signal processing and pattern recognition. Although computers were used, the ordinary methods of Computational Linguistics were not. The examples given here show how linear filters may be constructed to play the role of grammars, using purely automatic methods.

Part Two is designed to answer questions and objections from a number of viewpoints. It begins with a discussion of the relationship between the present approach and generative grammar. To emphasize their differences, the theory being presented may be described as <u>linear filtrative grammar</u>. Adoption of a filtrative model is possible without accepting linearity. However, even if one imagines nonlinear filters, there are advantages to using a linear space to analyse them.

Part Three is devoted to conclusions. First is a section discussing conclusions that could be drawn from the scattergrams produced by analysis of text from various languages. Following this are general conclusions and a summary of the argument.

PART ONE

A New Approach to the Study of Human Language

1.1 Introduction

The following pages contain a theoretical account of human language and a report of the results of various experiments with computer analysis of text samples. But this thesis is not just about language, it is concerned with the distinction between linear devices and nonlinear devices, their respective capabilities, and the notion that the human brain is a linear device. The distinction between linear and nonlinear systems is familiar to any mathematician, physicist, or engineer, but to few linguists.[1] For most of the former, this is a pragmatic issue, since nonlinear devices pose many difficulties, which are most often met by linear approximation.

I suggest that the human brain can be considered as a linear system, and that the best way of analysing human language follows from this assumption. The use of linear models for the brain is not a new idea, but in the past this has been treated as a methodological necessity, not adopted as a theoretical position. One reason for accepting linearity as an essential property of the brain is just that human beings are able to understand and cope with each other: this would be impossible on any other assumption.

---

[1]The terms 'linear' and 'linearity' are unfortunately used by linguists to mean something entirely different. I feel the mathematical and scientific usage has priority and will use these terms only in that sense.

Against this view there are a number of objections, which fall into two broad classes:

1. Mathematicians, physical scientists, and engineers are conscious of the severely resticted capabilities of linear devices, and may have trouble understanding how a linear device can be as powerful and flexible as the human brain.
2. Linguists and scholars from the humanities are usually unaware of the important advantages of linear devices, and the difficulties of analysing nonlinear ones.

The following pages include explanations and arguments aimed at objections from each of these classes. With the aid of examples it will be shown that nonlinear devices are particularly appropriate for applications involving calculation, while linear devices are most suitable in the area of communication, including simple encoding and decoding, summarizing and paraphrasing, classifying, and the recognition of patterns.

Linear devices are by far the easiest to understand, and have familiar and well-established mathematical properties. In contrast, human language has always been a mystery. By treating language as the product of a linear device, the present work aims to place the study of human language upon a firm and familiar mathematical basis, so that well-known mathematical methods can be applied in computer programs.

An important part of this task is finding some way of converting the sounds of speech or the letters of printed text

into more familiar mathematical objects such as numbers or vectors. This is a problem of interpreting symbols, and we may think of the meaning or interpretation of symbols as their position in some abstract space of many dimensions. An utterance, text, or string of symbols can then be treated as a path or trajectory in this abstract space.

On this analogy, the positions of each symbol in the space can be arrived at by noting the frequencies with which the different symbols occur adjacent to one another. Algorithms based on this method will be given, together with examples of the results. These algorithms were used to provide information about the sounds represented by the letters of the alphabet: information derived solely from their relative frequencies of co-occurence. This information was subsequently used by other programs that performed a simple low-level grammatical analysis.

One of the experiments to be discussed in a later section used the coordinate values obtained from an analysis of co-occurence to create a filter which could serve as a test of acceptability. The experiment involved a simulation of the phonological syntax or phonotaxis of a language, using a simple data compression scheme. A linear transformation was designed which compressed a vector of 12 elements into a vector of 9 elements. The pseudoinverse transformation was also created, which took a vector of 9 elements into a vector of 12 elements. The combination of forward and reverse transformation was a single linear transformation which I call a pseudoidentity

transformation, since it behaves exactly as an identity transformation for some vectors, but not for all.

In the experiment performed, the vectors represented strings of letters forming English text, and the program embodying the linear transformation would thus pass ordinary English text through unchanged, while altering any other input. By comparing input and output, the program could function as a simple recognition machine for English text.

A significant point is that the specification of the pseudoidentity transformation used in this device required only that certain programs be given a sample of English text. Without further information they were then able to construct a transformation that would pass the given sample, and because of the statistical similarities of all English text, would also pass most other samples of ordinary English. This automatic production of the phonological (or graphemic) component of a grammar can be seen as partly satisfying one of the ultimate aims of the Bloomfieldian era in linguistics: automatic grammar writing.

There are reasons for believing that similar methods could be applied at higher levels of linguistic structure, such as the arrangement of words within a sentence. I suggest that this result can also be extended to the production of grammars for parsing an utterance into its structural components, showing their interrelation.

1.1.1 Terminology

This thesis is about linguistic processes or operations, and about real or imaginary devices for performing them. Systems of equations are used as a mathematical description of the operations performed by a device, and the term 'system' can be used as a convenient way of referring to either a physical system or a system of equations. We may assume that each device can be described by many different systems of equations, and that any system of equations can be realized by many different devices. For this reason it is useful to adopt the mathematical practice of treating various devices and systems of equations as equivalent if there is an isomorphism between them. Thus we may say that two devices are equivalent, meaning that they are isomorphic, or even say that they are identical up to isomorphism.

The isomorphism between physical systems and systems of equations is easily made if the equations are first transformed into a particular form: each variable corresponding to an observable output should appear by itself on the left of the equal sign, and each variable corresponding to an input should appear as part of a mathematical expression on the right side of the equal sign. The system of equations can then be used as a model of the physical device or system by assigning a particular set of input values to the variables in the expression on the right side of the equation, performing the indicated mathematical operations, and treating the result as values of

the output.

This direct mathematical modelling of a physical process using known input values and finding output values is an important application of the mathematical model, but it is not the only one. It may be that the input to a process is unknown, but the output can be observed. In this case the mathematical model may or may not be useful, depending on the type of equation it involves.

In studying language, we are often facing this type of problem: we have a sample utterance, and want to know what the speaker intended it to mean. Thus we are given the output of a process and want to find the corresponding input. Neither the behaviourists nor the followers of Noam Chomsky have dealt with this possibility: the former have denied the reality of meaning, and the latter have taken it for granted. Both approaches seem misguided. Prior to either should be an investigation of models for which the meaning is treated as an unknown input to a process.

To find the input of a device from an observed output we must rearrange the equations so that the variables representing inputs appear alone on the left side of the equals sign. This rearrangement is exactly what is meant by solving the system of equations. Unfortunately, there may not be a unique solution. If a system is solvable, and has a finite number of solutions, the number of solutions is called the <u>degree of the system</u>. Linear systems are systems of the first degree, and if solvable, they

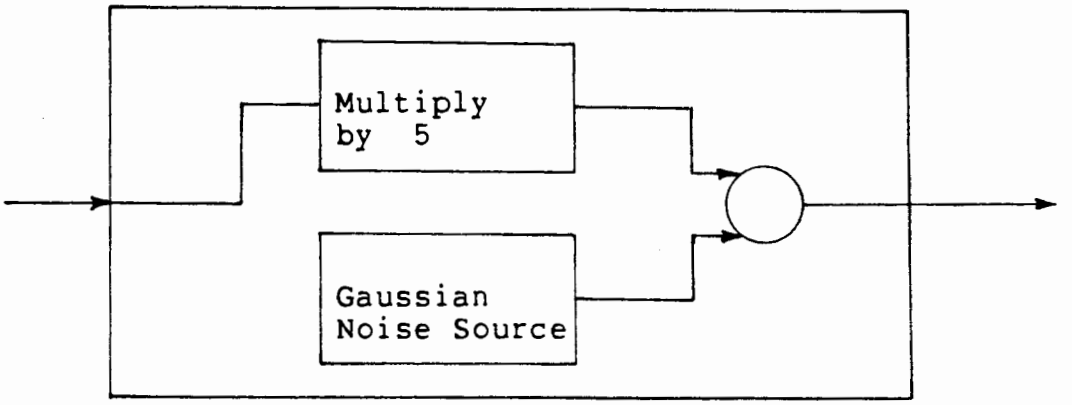
have a unique solution.

Throughout this thesis I use the term 'linear' and its derivatives, but I think these terms are confusing and should be replaced.

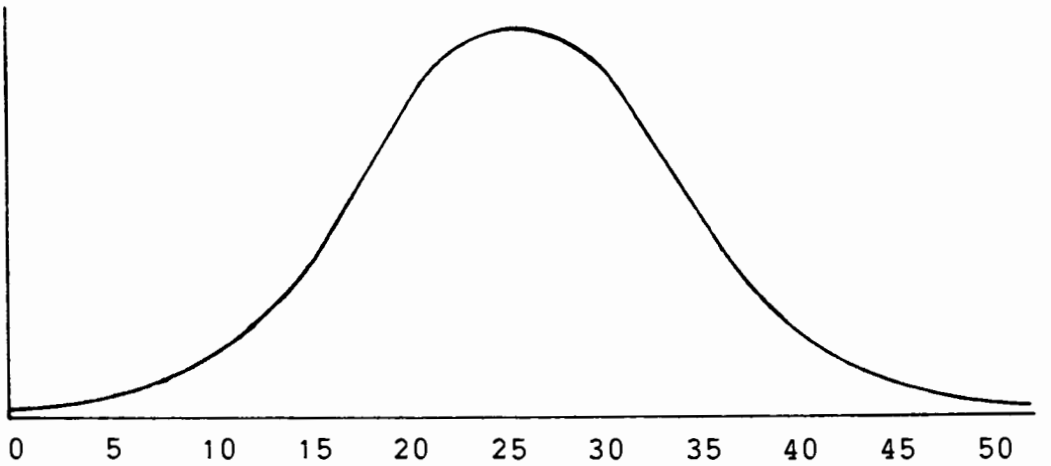
The laboratory investigation of such devices is based on the known properties of linear and nonlinear systems. A typical approach is to consider the device under analysis as a "black box" that cannot be opened. The usual problem is this: suppose that the experimenter can apply any inputs that he wants to the device, and observe the outputs. Can he come up with a description of the device that would be adequate to enable him to accurately estimate the input corresponding to subsequent outputs in the continued operation of the device?

Note that this is not the same question as whether or not he can describe the device adequately to predict future outputs. Both linear and nonlinear devices may be completely deterministic so that input completely determines output. Or, either type of device may be less than predictable because of internal noise.

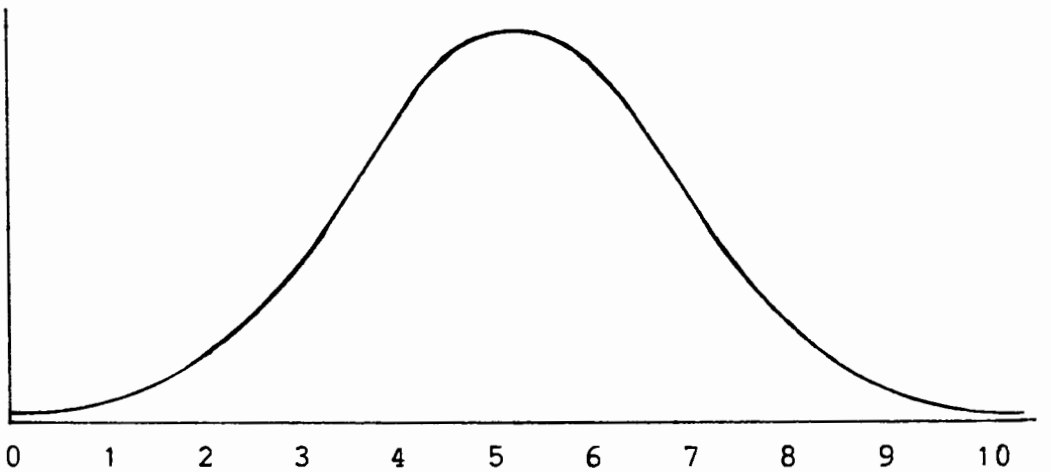
To avoid giving any impression that the theory to be described is deterministic, let us consider a noisy system. Given an input known exactly, the most that can be expected of a description of the device is a graph of probable outputs. Figure 1 shows a simple experimental situation. The device contains a source of noise, which is added to the output. If an input of exactly five volts is applied, and the output measured at



a) Linear Device with Internal Noise



b) Measured Output Voltage with 5 Volt Input



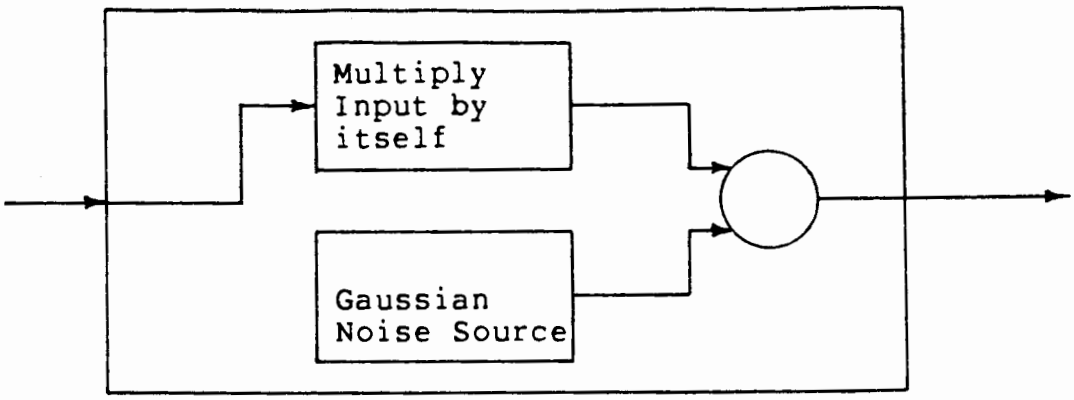
c) Estimated Input Voltage

Figure 1. Linear Case

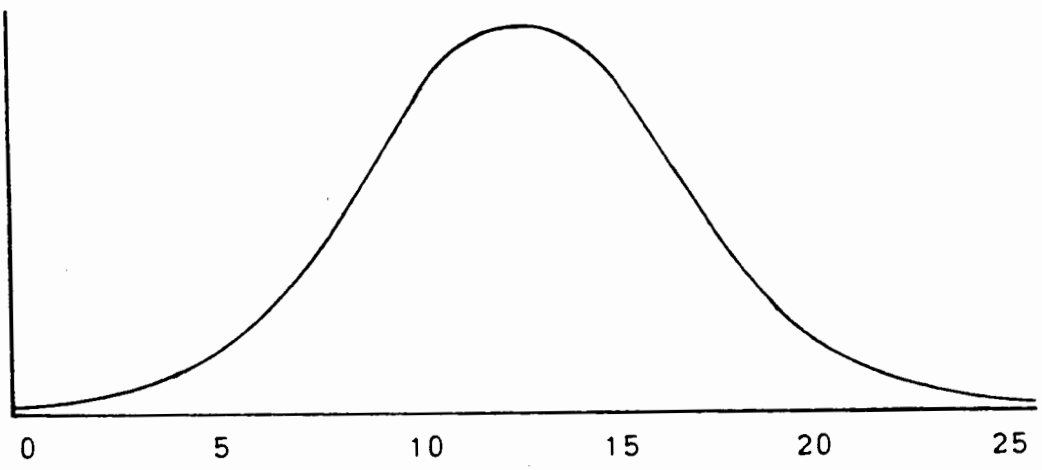
various instants, we would expect to see a graph such as given in the diagram. If the device being investigated is linear, as in this case, a similar graph can be given for the reverse problem. input from a measurement of the output. The second graph on the diagram is identical to the one above it, indicating that the problem of finding the most probable input corresponding to a measured output value is similar. of predicting the output from a knowledge of the input.

For nonlinear devices the reverse problem is quite different, as Figure 2 illustrates. The curve with several peaks shows the different situation that will normally hold for the reverse nonlinear case. Here there is no single most probable input, but several, each at the center of a range of possible inputs. The second graph, representing probable input estimated from a measured output will in general be quite different in appearance.

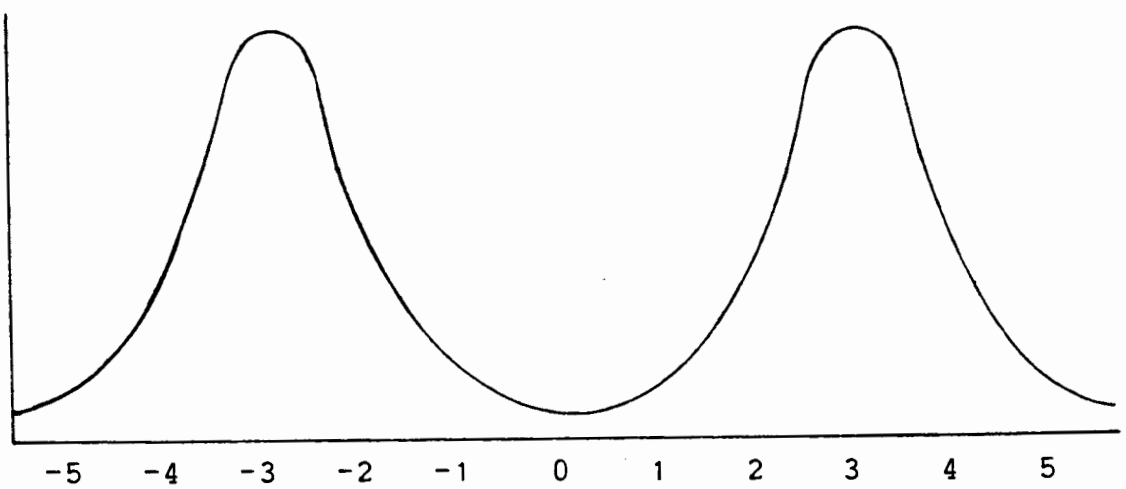
The number of peaks in the second graph for the nonlinear case is an indication of the degree of the system of equations describing the device. In general there will be n peaks for an equation or system of equations of degree n , although two or more peaks may be superimposed. Linear devices have a single peak since they are devices of degree one. If there is only one peak, then there is a single value that can be described as the most probable input, even though there is a range of possible inputs that could have produced it. The range of possible inputs forming each peak on the graph can be described as a lack of precision or as vagueness.



a) Non-Linear Device with Internal Noise



b) Measured Output Voltage



c) Estimated Input Voltage

Figure 2. Nonlinear Case

1.1.2 Order and Context

The term 'linear' is confusing but I have used it throughout this thesis because it is standard in mathematics. The term arises from the shape of the curve describing a functional relationship between two variables. If we assume the domain and range of the function is the field of real numbers, this terminology is appropriate. But mathematicians now apply the term 'linear' to similar functional relationships between variables defined over fields other than the real or complex numbers, and graphs of these functions are not necessarily straight lines.

I would suggest that a better term for use within mathematics might be 'pseudo-invertable'. This is clearly a weaker property than true invertability, and defines a class of algebraic structures which includes the ps class of groups as a proper subclass.

From this account it should be clear that linearity is a desirable property for linguistic processes, since we want to be able to find the unique meaning of an utterance, if it has one.

The linearity of linguistic processes may thus be related to the question of amibiguity in language. Early attempts at machine translation are often said to have failed because natural language is ambiguous. Words are usually assumed to have a number of different meanings, and no algorithm could reliably

choose amongst them. But there is another explanation for this failure: word meaning is very context-dependent. It is difficult to store enough information about the way meaning depends upon context, and to persuade a machine to use it correctly. Ambiguity and context-dependency seem similar, and many writers have used the terms as loose synonyms, but from a mathematical standpoint they are very different. Abiguity is a question of degree and context-dependency is a question of order. To attribute the failures of machine translating projects to the ambiguity of language is to confuse these important mathematical concepts. This probably happened because the methods of computational linguistics do not involve explicitly formulating samples of text as mathematical functions or equations. The account given here suggests that it is not necessary to use explicit mathematical formulations; we can find out the properties of possible formulations without actually creating them. Among the most important of these properties is linearity.

The major hypothesis behind all the research to be presented in this thesis is that human language is not ambiguous, even though its interpretation is highly dependent on context.

The mathematical formulation of this hypothesis is a statement about the human language processing facility, considered as a formal device:

> The human language processing facility is a linear device of high order.

In speaking of the order of a device, we are referring to a property dependent on the number of integrators, delay elements and/or memory locations it contains. The effect of these components is to make the output of the device depend not only on the current input, but on other recent inputs, or in other words, on the current input and its context. For systems of equations we do not speak of memory locations or delay elements but rather of differential and integral operators. In the theory of differential equations or difference equations the term 'order' reflects the maximum number of times the differential or difference operator is applied to a variable.

Figure 3 illustrates the range of possible systems. The rectangular hyperbola is an arbitrary boundary, intended to show how increasing order and increasing degree affect our ability to understand and use different systems. Systems represented by points between the coordinate axes and the curve are within the range of human understanding, while those beyond the curve in the right hand corner of the diagram are too complicated for any application.

In order to illustrate the use of this terminology, we may consider two examples. Although the class of nonlinear devices includes devices of every order, it is worth considering a non-trivial device of the lowest order[2] Consider a device that could be used for simple astronomical calculations, including

---

[2]Different branches of mathematics seem to differ over whether the lowest order is zero order or first order.

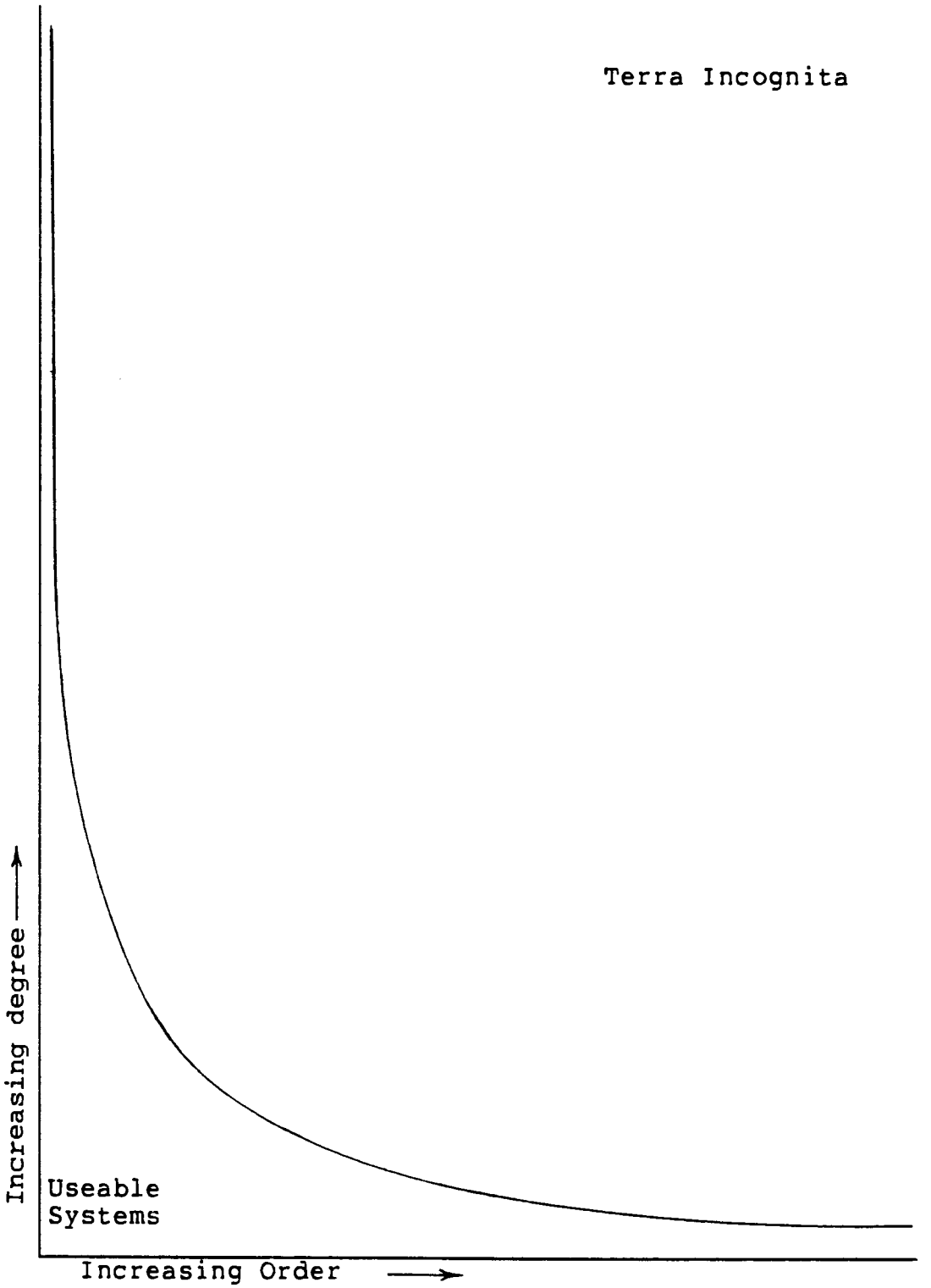


Figure 3. Classification of Systems

the prediction of solar and lunar eclipses. Suppose that the input to this device is an electrical voltage representing the current date and time as a single value. The output of the device is also a voltage representing a value for the angle between the solar and lunar discs as observed from a particular spot on earth. If the input is increased regularly, the output oscillates back and forth over a range of values, so this device implements a function of one variable which is clearly nonlinear. For use as an eclipse predictor, we would be interested in the problem of finding the inputs corresponding to a particular output: the one representing an angle of zero. Obviously there are infinitely many inputs that could produce that output, since there is no limit to the number of eclipses which have occured or will occur.

Note that with this device context is unimportant. A given input value will produce a given output regardless of what other inputs precede or follow it. Because the device evaluates its input with no consideration of context, it may be described as being of zero order. But the infinite number of different inputs that could produce any particular output indicates that the device is of infinitely high degree.

Next we may consider a device of first degree, but of arbitrarily high order. It is well known that the propositional calculus of logic can be represented by many different notational conventions. Let us envision a translator between two of these: fully bracketed infix notation and Polish notation. In

the former each operator occurs between its operands, and parentheses are placed around each compound term:

(a&b)V(c&d)

In Polish notation operators are placed to the left of their operands and no brackets are used:

AKabKcd

To make this example more comparable to the previous one, let us consider that the letters and symbols used in these notations are encoded as they would be by a computer terminal with a modem, for transmission over a telephone line, and that the translating device in question is to be placed between two such terminals. Thus, this device also has as a single variable voltage as its input and as its output.

If one were to sample the input and output voltages at a variety of times and tabulate them, the device would seem to be highly ambiguous, since a given output voltage will occur at the same time as an apparently random selection of input voltages. But indeed, the device is not ambiguous. If the output voltage is sampled regularly and frequently for a long enough period of time, it can be decoded, and will then be seen as a representation of some expression in Polish propositional calculus. If so, the input that generated it could be calculated quite accurately.

If we ignore noise and indeterminacies in the computer terminals and think only of the relationship between the input symbols and output symbols, it can be seen that a particular

expression in the output corresponds to one and only one expression in the input. But the complete expressions must be examined to be sure of this correspondence. From the account of ambiguity and context given previously it follows that this device must be a linear one of very high order, even though we cannot give equations to describe it. If there is no limit to the length of expressions or the number of variables, the device would have to be of infinite order.[3]

How can we be sure the above device is linear? As will be shown below, linearity is related to invertability, but is a weaker property. If a device is invertable it is linear, although the converse does not hold. In this case, we could also describe a device that would perform the inverse translation from Polish notation into fully bracketted infix notation. Since each of these translators acts as the inverse of the other, both are invertable and so both are linear.

It may be objected that known non-linear operations are used in computer programs implementing similar devices. This is not a problem, since it is known that the combination of two or more non-linear devices into a single device may yield a linear device. The converse does not hold: linear devices combined with other linear device can only produce another linear device.

---

[3] In practice such a device would be limited in some way and realized with a finite state machine. Linear devices acting as finite state machines are discussed in a later section.

1.1.3 Precision and Vagueness

Vagueness, or lack of precision, is a normal and generally desirable property of all expressions in any language including formal languages. Rather than having one precise meaning, a word usually has a range of similar meanings. The term 'animal' is vague, but useful. To be less vague, one can say a 'motile, food-consuming, life form', thus defining the class of animals. Alternatively, one could specify a subclass, or perhaps a single animal, such as John's white Scotch Terrier. With either approach, some vagueness is left, but there is much less.

The concept of vagueness is essential to an understanding of the range of abilities of linear devices. The previous example was a hypothetical translator, and was invertable. Some linear devices are not invertable, but they still have a unique pseudoinverse, which is the best possible approximation to an inverse. For illustration, let us consider two hypothetical linear devices, a "summarizer" and an "expander". Neither of these devices is truly invertable.

A summarizer could be described as a device for abbreviating a long text. Given a book length document, it would produce a short paper, or perhaps an abstract of it. Obviously the output of this device cannot contain as much information as the original. One method of abbreviation would be selection of a short part of the book, but to live up to its name the device should produce a summary that includes all sections of the original. Since it must include less information than the

original, while not omitting any section, its treatment of each section must be poorer than in the original. The shortened versions of chapters, paragraphs, and statements will have to be less precise than before, but this does not mean they will be more ambiguous.

Looking back at Figure 1(c) we may imagine that this graph represents a range of interpretations for some statement in the original document. The single peak represents the most likely interpretation, and the width of the peak indicates the amount of vagueness. A summary of this statement will be a shorter statement, or perhaps a word, but its range of meanings should be a broader or flatter curve, indicating less precision. The range of interpretations for the summary will be a broader peak, but it should still be a single peak, not a double peak (or bimodal distribution) as in Figure 2(c). It would be wrong for a summarizer to introduce ambiguity into the original, but vagueness is unavoidable. According to my account of these terms, this would make a summarizer a linear device.

A linear expander may also be considered. One way of describing such a device would be as one that would replace uncommon terms in a document by a definition or paraphrase. However it works, such a device can be envisioned that would come very close to inverting the output of the summarizer just described. If we are satisfied that a summarizer is a linear device because it does not introduce amibiguity, then we can be sure it will have a pseudoinverse, which will come as close as

possible to inverting its output. Again, such a device should not introduce amibiguity into its output, nor should it eliminate ambiguity that it finds in its input. But a longer treatment of a subject can be expected to be more detailed than a shorter one, and so this device may reasonably reduce the vagueness found in its input by estimating the most likely meaning of laconic expressions.

One may wonder what would happen if the summarizer and expander were both applied, in that order. Ideally, the original document would be reproduced unchanged. More likely, however, the output would resemble the original but be blander, with surprising usages replaced by more conventional ones.[4] This is the first example of a kind of filter that could be called a pseudoidentity filter, since its effect on certain (very unoriginal) input texts would be indistinguishable from the identity filter (or trivial filter) that passes text through unchanged. In a later section I use such a device as a model for the grammar of a language.

It must be emphasized that the hypothetical summarizing and expanding devices should be linear, neither introducing nor removing ambiguity. This is true of any device for processing language. A machine translator should also have this property, since the translation of a text should be exactly as ambiguous

---

[4] Kohonen(1977) describes a "novelty filter" designed to remove familiar material from its input, leaving only novelties. A filter created by combining summarizer and expander, would be just the opposite: it would remove novelties, replacing them with more familiar usages.

as the original, but in a different language. Whether or not this is possible is not immediately obvious, but it could serve as an ideal standard to be approached by actual translating devices. The theory that linguistic devices are linear devices is not a tautology, since perfect translation between human languages may not be possible, and in particular some introduction of ambiguity may be unavoidable.

In the Romance languages there is no neutral pronoun corresponding to the English word 'it'. Suppose we read English translation including the following sentences:

'I like your new table, and that is a lovely bowl on it.'
'Where did you get it?'

and the reply:

'He gave it to me.'

We don't know if the word 'it' refers to the table or the bowl. In the original language the pronoun corresponding to 'it' will show the gender of the object, and thus may identifify it.(In French, 'le' or 'la', for 'le bol' or 'la table', for example.)

This would have made the statements unambiguous. In this case the translator could be criticised for introducing ambiguity, and we can see how it could have been avoided. It is not clear that it is always possible to avoid ambiguity in translating between natural languages.

We have considered three kinds of devices: summarizers, translators, and expanders. These basic units, and any combination thereof, could all be described as devices for communication, as distinct from calculation. Linear and nonlinear devices each have their own area of application, and sensing or communicating information is best performed by linear devices, since the output of these devices may be described as a translation, expansion or summary of their input.[5] Given the output of a linear device we can recover its input with minimal error, and this is a basic requirement for any channel of communication.

The suggestion that human brains are linear devices, although very complex ones, emphasizes our role in sensing and communicating information, at the expense of our role in processing information. This is a very different view of human abilities from that common in most academic disciplines. For example, researchers in the new field of Artificial Intelligence commonly use a model that emphasizes processing abilities,

---

[5]It must be noted that the term 'translation' is used in the theory of operators to refer to a nonlinear operation. In this case I prefer to ignore the mathematical usage and use the term translation to refer to the (linear!) operation of translating between languages.

including supposed abilities that seem very much like nonlinear operations. Indeed, the AI community seems willing to use any operations at all to achieve their ends, even if the resulting models bear very little resemblence to known features of the human brain. Only occasionally does one find a suggestion that perhaps more could be accomplished with less, or that there might be some point in limiting the operations employed to some extent.

1.1.4 System Identification Techniques

The notion that human beings are linear devices immediately suggests a number of routine mathematical techniques such as are employed in systems engineering. The first steps in dealing with a linear system of unknown properties are known as system identification techniques. There are a variety of different methods that can be used, depending on what kind of information is available about the system.

Most methods of system identification involve applying a known input to the system, observing the output, and applying mathematical techniques to the comparison of input and output. The methods of word-association testing used in psychology could be described as system identification, with the stimulus word being a known input, and the response the output of the system.

Word associations tests do involve the human language processing facility, but only in a limited way. There are many other ways of defining an experimental situation where a given

input can be compared with the response it produces, and one of great interest is the study of simultaneous translations, in which the translator is under pressure to respond as quickly as possible with a translation of whatever he hears.

Machine translation has often been attempted, but success has been limited by the supposed complexity of human language. The arguments given above suggest that however complex human language processing is, it should be a linear operation. If this is so, certain very powerful system identification techniques apply, particularly spectral analysis.

The spectral analysis of linear systems makes use of integral transforms such as the fourier transform. Each particular sample of input can be transformed into a complex frequency spectrum with the fourier transform. The spectra of many different inputs can be averaged to yield a characteristic spectrum of the input, and the same process can be applied to yield a characteristic spectrum of the output. If the system is linear and certain other conditions exist, the ratio of these is the characteristic spectrum or system transfer function of the device. Once the system transfer function of a device is uncovered, the system is completely identified and its output for any other input can be predicted.

The idea of going through this series of operations and arriving at a system transfer function for the process of translating from one language to another was very exciting, and has served as a stimulus for much of this research.

Translation from one language to another is often thought of as two processes combined: the decoding of an utterance in one language, yielding its meaning, and the subsequent reencoding of this meaning in the second language. If these two processes represent definite subsystems it may be possible to isolate them. If we could do this we would have systems for translating between an ordinary human language and a neutral "language of thought" or "logically perfect language".

The existence of these systems has often been discussed, particularly by grammarians of the school of Generative Semantics, for whom the grammar of a language was a system of rules for translating a logical representation of meaning into the correct form for expressing that meaning in the language. System identification techniques do exist for isolating subsystems that cannot be directly observed. If there are such devices, one way of finding them would involve examining known devices with similar properties. This is the essence of a systems identification technique which begins by applying random input to a device with externally variable characteristics, and then adjusts the device until its output matches that of the system being investigated. For linear systems this method is valid, with matching outputs implying similar characteristics. An attempt to find a linear model of grammar could thus begin with an attempt at synthesis: we could try to find or construct a linear device whose output is similar to human language.

Oddly enough, a concrete model of one such process has been available for decades. Markov devices whose output approximates a natural language are usually represented by probability matrices, in which each entry represents the probability of transition from one internal state to another. In moving between states the device is usually made to output one or more symbols, and is therefore treated as a generator of strings of symbols.

This treatment of Markov devices is misleading, since it hides the input. A device with no input is difficult to analyse, and cannot even be labelled linear or nonlinear. In any actual implementation of a Markov device, the input is a sequence of random (or pseudorandom) numbers from a random number generator. A Markov device as actually implemented is thus a filter which takes an uncorrelated input and imposes some characteristic form or structure upon it.

Markov devices played a central role in the development of modern linguistic theory after being proposed as part of a model of a grammar by C. F. Hockett. One of Noam Chomsky's earliest and best known contributions to linguistics was a demonstration that "finite-state Markov sources" were inadequate as a model of grammar. I believe Chomsky's account of Markov models is irrelevent and dangerously misleading. If Markov devices are treated as filters rather than generators, their true importance can be seen, and their limitations overcome. In a later section I show how linear Markov filters can be used as a model for grammars. This does not contradict Chomsky's result, since the

'Markov sources' he discussed were defined differently, and his notion of grammar was different. Chomsky's mathematics is not wrong, but his definitions are too narrow.

Markov devices can be implemented as linear devices which filter the output of a random number generator. If we accept the linearity of human language processes we should then attempt to understand linear Markov filters, particulary ones of high order, since their output is so similar to ours.

One problem with high-order Markov devices that has limited their application is the large amount of data they must employ. A first-order Markov device requires a table of at least 729 values, representing the probabilities of each letter (including the space between words) occurring after each other letter. A second order device requires four times as much data, since pairs of letters are used. Thus the amount of data required grows as the square of the order. There are two possible attitudes to this problem:

1. These requirements are unreasonable: we should use data compression or some better algorithm to reduce the amount of data required and/or the rate of growth in required data.
2. These requirements are unavoidable: we must accept the data requirements as an inevitable consequence of the complexity of human language, and try to find out what function complexity serves.

The first of these attitudes seems justified, and in fact I have simulated first and second order Markov devices using tables

containing fewer than one hundred data values. Unfortunately, it is not so easy to extend the methods used in doing this to higher-order devices. To create high-order filters that simulate human grammars cannot easily be done with compressed data, and this reinforces the notion that human language is very complicated.

One central issue in all debates about language is the question of the total complexity of a human language. Chomsky has argued that language is so complicated that knowledge of it must be innate, while others have argued that it must be simple enough to be learned. One of the first tasks in the application of systems identification techniques must be the estimation of this complexity.

Simplicity and information content are related, since we call a system simple if it may be described in a few brief sentences, but complex if a lot of information must be given for any adequate description. Human languages are considered to be very complicated, since vast amounts of information must be given in any usable account of a language. Indeed, it is doubtful if any natural language has ever been completely described.

One problem that has not been dealt with by Chomsky is why language is not much simpler. Esperanto is an artificial language which was intended to be much simpler and more regular than ordinary natural languages. But Esperanto is not as simple as it could be, since Zamenhof borrowed the vocabulary from a

number of natural languages. The lexicon of Esperanto is complex by comparison with other artificial languages that have been proposed, which have employed a very small number of morphemes and simple rules for combining them. Such systems are actually used in libraries as classification schemes. The Dewey Decimal system, for example, uses 10 morphemes, represented by the integers from 0 to 9, and forms expressions representing an enormous number of concepts by combining them.

Very simple schemes along these lines have been proposed since the early 16th century (Knowlson 1975) and later by Descartes, and by Leibniz. Specific versions tailored more to the needs of people, rather than as philosophical ideals, have been examined by various European scholars over a period of 500 years, but none of them has come anywhere near even the limited success of Esperanto.

If language is complicated, its complexity may serve a function. It has been suggested that language contains cultural information, and is an essential repository of human knowledge.

1.1.5 Extraction of Information from Language

The idea that language has content appears with the discovery of Sanskrit and the rise of historical philology, culminating in the writings of Wilhelm von Humboldt. The correlation between sound changes in language and historical movements of people was often discussed in the 18th century, and the idea of extracting information about historical events from

language aroused considerable excitement amongst the romantic philosophers and philologists.

It is in the work of Humboldt that this theory is most fully developed. In his introduction to language, prefaced to a serious work in the comparative linguistics of Indonesian languages (Humboldt 1836, see also Miller 1968), Humboldt argues for the complete relativity of language to culture. For him, language thus does not have any arbitrary components, but is precisely determined by national culture. Since the main argument for the arbitrariness of language is the supposed existence of more than one language, in denying arbitrariness Humboldt does indeed suggest that there is only one language.

In fact, Humboldt argues, it would be equally correct to say that there is only language, or that each and every person speaks a different language. Which usage is to be preferred depends on whether one uses 'language' to refer to the medium in which thought is expressed, or the specific reflections of thought in the speech of a single individual. Throughout the rest of his treatise Humboldt adopts the latter usage. He speaks of a multitude of different languages, each one capturing and preserving the culture of some nation or individual. Humboldt clearly advocates the preservation of differing national languages and even the cultivation of differences. At one point he writes that it would be better if each individual spoke a completely different language which accurately mirrored his own thoughts, regardless of the disruptive effects of this on human

communication. If each language contains a fixed amount of information, it would follow that if more languages are studied, more information could be obtained. If each person spoke a different language, the total information available would be enormous.

This seems very far from Leibniz's dream of a single ideal language for all mankind, but remember that Humboldt allowed that 'language' could also refer to the medium in which thoughts are reflected. It seems that Humboldt incorporated the ideal language of Leibniz as a medium of expression, and went on to consider the culture expressed in it.

Humboldt does give some characteristics of this medium, and its properties are wonderfully logical. He speaks of a 'true morphology' of language, in which words are formed from meaningful sounds, with the stipulation that "Since words are always juxtaposed to ideas, it is natural that related ideas are designated by related sounds". (Humboldt 1836:55ff) In developing his theory of language Humboldt shows again how natural languages deviate from this ideal. But he makes this deviation functional; he claims that it expresses national culture.

This is a remarkable view which should be expanded upon. A person may suppose that he or she is expressing some meaning, represented by the small letter, 'a', or another,represented by the small letter 'b'. We may think of these meanings as being expressed in some language, which might be described as a

function that takes the meanings into the sequence of sounds forming an utterance. Thus one may say that to express these meanings a person utters F(a) or F(b). Another person speaking a different language may express the same meanings by uttering G(a) or G(b).

However, if Humboldt is right, each person is expressing more than just the meanings he is conscious of. Each person expresses some of his own cultural background as well. If we use two more small letters, 'f' and 'g', to represent the cultural knowledge of the respective speakers, then we could say that the first person is really expressing the sum of the intended meaning and the cultural knowledge, which could be written 'a+f' while the speaker of the other language is expressing the same intended meaning together with his different cultural knowledge, which can be described as 'a+g'.[6] Both people could be speaking the same universal language, represented by the function U. Thus one person says U(a+f) or U(b+f) while the other says U(a+g) or U(b+g). For Humboldt this is an equivalent way of describing what happens when two people speak differently.

If we use lower case letters to represent statements or meanings, and upper case letters to represent functions, then Humboldt's view can be expressed by saying that F(a)=U(a+f) and G(a)=U(a+g).

---

[6]The plus sign (+) in these examples must not be taken as ordinary addition, but rather as addition of vectors in some metric space to be defined later.

In interpreting this it is important to treat 'f' and 'g' as variables depending upon the whole of each person's past input, rather than as constants. We should also treat 'a' and 'b' as input variables, since we are describing the human language processing facility, and intended meaning is an input to that facility. If these conditions are observed, the equations given above serve to define the functions 'F' and 'G' as linear functions of the variables. Thus, this formulation suggests a linear model of language such as is being presented here.

Humboldt's work represented an expansion of the idea of recovering historical information from language to include all of the cultural knowledge of a society. In Humboldt's view language is complicated because it has so much content. But what, specifically, is this content, and how exactly can it be recovered?

A possible answer is provided by the methods and results of classical scholarship, where fragments of vanished texts may be recovered from existing texts of subsequent authors. A thorough knowledge of the style of the latter, aided by statistical analysis, can reveal words and phrases, that are probably quotations, and these can sometimes be pieced together from several sources to yield a reconstruction of what the earlier author may have written. Classical scholarship is a very difficult area, and is only used here to show that historical information can be extracted from language, if enough time and energy is applied. This historical information may be nothing

more than samples of text from an earlier period.

The problem with classical scholarhip as an example is that it still depends heavily on the knowledge and intuition of the scholar, producing few results from a lot of labour. This would be an ideal area for complete automation, in which a large computer could digest and analyse masses of material, if only some way could be found of making written language more accessible to the computer. These methods could then perhaps be extended to the study of more recent history, and some of the cultural knowledge of societies that so interested Humboldt could be extracted.

1.2 Computational Development of a Linear Model of Language

The possibility of synthesizing a linear model of human language processes remains as an ultimate goal for research based on the linear model of language processing.

The problem of dealing with human language is that the data is not immediately available in a form convenient for mathematical analysis. It was necessary to find ways of submitting language samples to a computer.

1.2.1 Mathematical Representations of Language

The experimental work began with an attempt to use digitized speech waveforms as data. The plan as first conceived was to digitize utterances of approximately one sentence in length, to record digitally many of these sentences, and to analyse the data using fourier transforms and related techniques. The problem with this approach was the extremely large number of data points that had to be collected to represent a single sentence. Even short sentences take about one second to utter, and in that time about ten thousand values would be collected.

Recording and analysis of that many data points proved difficult, so research was then conducted in an attempt to apply some relatively simple data compression techniques. This was partially successful, but in the absence of special purpose hardware still required the recording of thousands of data

values prior to their application, and this was extremely difficult with the available equipment.

The problems of working with digitized speech could be avoided altogether by using a symbolic representation of speech in orthographic symbols, if enough samples from different languages in the same notation could somehow be obtained. It is easy enough to obtain ordinary text samples from various languages, but difficult to collect adequate phonetic representations of speech. If samples could be obtained and entered into the machine, the symbols used could then be translated into feature vectors using the values already chosen by generative phonologists.

The idea of borrowing distinctive feature values from generative phonology suggested a consideration of other aspects of generative phonology. Chomsky and Halle's discussion of the phonology of English shows an unusual respect for ordinary English spelling and orthography, which are shown to reflect the underlying phonetic forms of English. If this is so, then perhaps ordinary printed text could be used instead of special phonetic transcriptions. By adopting Chomsky and Halle's feature values for phonemes represented by the alphabetic symbols it would be possible to convert ordinary printed text into a form suitable for further analysis. Methods of factor analysis or feature extraction could then be applied to these feature vectors to produce vectors that are more suitable for mathematical treatment.

It was supposed that each symbol would be eventually represented by several numbers, that is to say by a vector of numbers, and that the representation of a sentence would therefore be in the form of a rectangular matrix. Such matrices would be similar to those used in generative phonology to represent sequences of speech sounds. Each vertical column represents a single speech sound, and their juxtaposition into a matrix is a convenient device for representing a sound sequence.

My first attempts at finding coordinate values adopted feature vector assignments from Chomsky and Halle, encoded text samples using them, and then applied factor analysis to look for a more convenient set of values. This was not hard to do for English, but presented problems in dealing with other languages. Although linguists have looked for a universal set of distinctive features, there is no agreement on one. Nor is there any agreement on the status of alphabetic symbols for other languages, which may or may not correspond to underlying phonemes and have an obvious representation as a feature vector.

Because of these problems it was decided to attempt a mechanization of the processes by which phonologist arrive at feature vectors. In studying these processes I found several interesting ways of representing text.

The difference between the representations that were developed in the course of this research and the feature matrices used in generative phonology is that the latter originate in the phonologist's personal intuitions, aided by

various arguments from the philosophy of science. One consequence of this process of ascribing features is the common use of binary values for features, although certain phonologists have argued for a wider range of values.

In the research discussed here feature vectors for alphabetical symbols come from a statistical study of the text, and may be positive or negative real numbers, not necessarily integers. The number of values required for each vector was not known, though some estimate could be made from a review of those phonologists who do believe in continuously varying features. It was originally supposed that about three or four coordinates for each vector would be adequate.

The methods of feature extraction used here are based on the least-effort principle of functionalist phonology. Functionalists believe that language is shaped by a principle of least-effort or economy. Certain sequences of sounds involve more effort than others, and are accordingly less favoured. Some sequences will involve so much effort as to be almost impossible.

Thus for the typical speaker of English the k - a - t sequence of /kat/ is an easy and natural motion of the articulatory organs, while the k - t - a sequences of /kta/ is quite difficult.

Phonetic features chosen to satisfy a least-effort principle will tend to have values which change as seldom as possible during the course of an utterance. One way of stating

this is to suggest that feature values have been chosen so that the probability of a pair of sounds occurring in sequence is proportional to the similarity of their feature vectors. If binary features are used, then counting changes between adjacent symbols would reveal their similarity; if continuous features are used, utterances would be represented by smooth paths or trajectories in some abstract space.

1.2.2 Finding Coordinate Values by Successive Approximation

The task of finding suitable coordinate values for the alphabetical symbols can be considered a problem of solving a system of equations. The system of equations to be solved is not given explicitly, but is implied by the principle of economy or least-effort.

If four coordinates were used for each alphabetic symbol, a 100 character length of text in a language would be expressed as four functions, each represented at 100 points by the values of one of the four coordinates over the whole length of text. One simple way of characterizing the desired properties of these coordinates is to say that the values for each alphabetic symbol should make the resulting four functions as smooth as possible, while still encoding the necessary information.

Given this principle, there is a remarkable method for arriving at suitable coordinate values for each symbol. The simplest version of this method involves choosing the coordinates at random and then passing the resulting functions

through a low-pass filter. The low-pass filter will replace the values of the function at each point with new values that are arrived at by averaging the assigned values with some weighted sum of the neighbouring values. Thus, the values of neighbouring symbols will more closely resemble each other.[7]

After filtering, the four parallel functions representing the text sample no longer have a single set of values corresponding to each letter of the alphabet, so the various values for each symbol will have to be collected together and averaged. The original random values for each symbol are thus adjusted by the filter to more closely approximate those of the symbols which are most usually found nearby. This filtering, collecting, and averaging operation can be repeated as often as necessary. Each time it is applied the values in the vectors representing each symbol become more like the corresponding values in its most common neighbours. The filtering process could be described as an artificial assimilation, in which the properties(i.e. coordinate values) of each symbol become progressively more like those of its most common neighbours.

The functionalist account suggests a historical process by which sequences of sounds are created through the novel combination of existing morphemes, and then become altered through the replacement of difficult sequences with new sequences that are similar, but more easily pronounced. The results of this continual process is a language in which most

[7]The following section on filtering shows more clearly how this operation takes place.

sequences of sounds are relatively easy to pronounce, but which contains a few new coinages that are still phonetically awkward. The original random assignment of values is like the formation of words by combining morphemes. Subsequent filtering parallels the phonological changes affecting words as assimilation makes adjacent sounds more similar.

A series of computer programs performed this filtering operation on whatever initial set of coordinate values was provided. The new values resulting from this operation were an improved set of coordinate values, in that sequences of such vectors better obey the least-effort principle. It was possible to improve an existing set of coordinate values by applying this filtering operation, and it was also possible to create a set of good coordinate values by starting with an entirely random choice of values and filtering them repeatedly. It should be noted that this process has an unfortunate side effect: the four sets of coordinates tend to converge to a single set of coordinate values. In order to make sure we recover four distinct sets of coordinates from the filtering process, it was necessary to 'push them apart', using the Gram-Schmitt orthonormalization algorithm after each iteration of the basic filtering operation.

The most interesting case for our present purposes is one in which the filtering operation is a simple moving average filter of order two. Such a filter would add some fraction of the assigned values of each occurence of a symbol together with

fractions of the assigned values of both the preceeding and following symbols.

The process of averaging the various new values assigned to each alphabetical symbol involves adding the values and dividing by the number of values. This addition involves a large number of terms, which can advantageously be rearranged. Rather than include the values of the letter R as separate terms in the calculation each time R precedes the given letter, all such instances may be expressed as a single term with a multiplying factor.

If we carry out this rearrangement, it is only necessary to count the number of times each letter occurs before and after each other letter. The simplest way to arrange the number representing the results of this counting is in a matrix. Using the ordinary definition of matrix multiplication this matrix can serve as a filter that is almost exactly equivalent to that specified above.[8]

It should be noted that the intermediate steps of counting symbols and multiplication by the resulting matrix can be hidden in the program so that from the user's point of view it is the text sample which serves to filter the vectors representing alphabetical symbols.

---

[8] The only difference between the filtering operation as first described and the matrix multiplication comes from the elements on the diagonal of the matrix which are not correct for this purpose. In this regard we may note that adjusting the diagonal elements of a matrix is a standard operation of factor analysis.

1.2.3 Relationship to Factor Analysis

Thus, an equivalent algorithm substitutes for the fourth step a multiplication by a suitable matrix. The process of repeated matrix multiplication and normalization of a single vector is a well known method for extracting the principal eigenvector of a matrix. Repeated multiplication by a matrix tends to turn an arbitrary input vector into an eigenvector. If a series of different vectors are repeatedly multiplied by a matrix, and are kept othogonal by repeated applications of the Gram-Schmitt process, then they are turned into separate eigenvectors.

If the above algorithm is allowed to produce a complete set of coordinates, instead of just four for each symbol, so that the number of coordinate values assigned to each symbol is the same as the number of different symbols, then the above process will normally produce an orthogonal set of feature vectors. If the set of coordinate values is considered as a square matrix, then the application of the Gram-Schmitt process to the rows of the matrix also makes the columns orthogonal, since any square matrix with orthogonal rows also has orthogonal columns.

This method of producing eigenvectors, using the Gram-Scmitt process to keep the vectors othogonal, tends to order the eigenvectors in order of the size of their corresponding eigenvalues. The first vector treated by the proces is merely normalized. Subsequent vectors are altered more and more drastically by the orthonormalization process, and

therefore tend to correspond to smaller eigenvalues.

If the matrix of counts is considered as an autocorrelation matrix for some unknown set of vectors, then the process of extracting its eigenvectors would be the familiar method of factor analysis or the Karhunen-Loeve expansion.

The above process is interesting since it arises from a simple consideration of a principle familiar to linguists. In the course of lengthy computational experiments the original clumsy method of translating a text sample into features and filtering was replaced by the simpler method that used a matrix of adjacent symbol counts. For a while this was employed in a filtering operation through matrix multiplication, but it was eventually realized that this operation was no more than factor analysis of the data in the matrix.

The use of a matrix of adjacent symbols counts as data is most interesting when it is remembered that a first order Markov device (as usually implemented) contains a table of adjacent symbol counts. Thus the method just developed amounts to applying factor analysis to the matrix contained in a Markov device. The results reinforce the view that Markov devices are linear filters and usable as a model of grammar. This will be discussed again in the last section, which gives examples based on analysis of several languages.

1.2.4 Adjacent Symbol Counts as Data

The fundamental type of data used in the first part of this research was obtained by counting the number of times each letter occurred before or after each other letter. A program created a matrix of 27 rows,27 columns. Each row of this matrix represented a different letter of the alphabet, including the blank. In counting continuous text, all punctuation was ignored, and multiple blanks replaced by a single blank. The program read some sample text and assigned one row of the matrix to each symbol in the text. For convenience in using text prepared for other purposes, the program identified upper and lower case letters, and treated all nonalphabetic symbols as blanks. Sequences of blanks or other nonalphabetic characters were compressed into single blanks. For most of this research the standard English alphabet of 26 letters was used, with the addition of the blank that indicates the break between words.

The result of counting could be shown in a table or matrix such as given in Table 1. In treating the matrix of adjacent symbol counts as 27 pieces of data about each of 27 items, statistical methods were employed to condition and describe this data.

The 27 data values in each row vector can be taken as defining the position of that letter in a 27 dimensional space. The statistical methods employed include methods of factor analysis, used to rotate and project this space onto a two-dimensional subspace that may conveniently be represented as a scatter diagram. (see Figure 9 page 62)

	-	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
	0	40	12	30	32	19	14	7	7	40	1	2	16	14	8	20	33	0	9	28
A	13	0	16	8	1	0	0	8	0	4	0	2	20	7	27	0	6	0	27	8
B	0	3	0	0	0	7	0	0	0	6	0	0	13	0	0	7	0	0	0	2
C	1	13	0	3	0	24	0	0	8	6	0	0	5	0	0	25	0	0	4	0
D	24	1	0	0	0	30	0	0	0	14	0	0	0	0	0	3	0	0	0	6
E	99	14	0	10	16	2	3	2	0	1	0	0	9	14	33	1	5	9	37	31
F	7	4	0	0	0	6	7	0	0	6	0	0	0	0	0	10	0	0	3	0
G	18	2	0	0	0	8	0	0	3	4	0	0	2	0	1	0	0	0	7	2
H	14	9	0	0	0	46	0	0	0	12	0	0	0	0	0	12	0	0	0	0
I	6	5	5	27	2	6	9	5	0	3	0	0	13	10	61	32	2	0	3	16
J	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	2	0	0	0	0	0	0	0	0	1	0	0	0	0	2	0	0	0	0	3
L	16	14	0	2	2	23	0	0	0	16	0	0	5	0	0	5	0	0	0	4
M	10	15	3	0	0	15	0	0	0	8	0	0	0	7	0	8	8	0	0	4
N	48	5	0	15	9	13	1	29	0	4	0	1	1	0	0	11	6	0	0	17
O	12	0	5	1	10	0	5	4	0	0	1	2	8	14	51	4	6	0	24	4
P	0	9	0	0	0	14	0	0	5	0	0	0	5	0	0	7	2	0	18	2
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R	23	23	0	0	3	34	1	0	1	12	0	1	3	8	1	12	2	0	4	6
S	69	1	0	7	0	14	3	0	4	19	0	1	2	1	0	5	0	0	0	10
T	46	11	0	0	0	33	0	0	65	49	0	0	1	0	0	11	3	0	10	8
U	0	10	1	4	5	6	0	1	0	2	0	0	11	4	7	1	0	0	2	9
V	1	9	0	0	0	14	0	0	0	12	0	0	0	0	0	1	0	0	0	0
W	1	1	0	0	0	3	0	0	3	6	0	0	0	0	1	3	0	0	0	1
X	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0
Y	17	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	4
Z	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Table 1.

Adjacent Symbol Counts

Several rows of the table given previously are reproduced again in Table 2.

The first set of five rows are for the vowel sounds; the second is for the labial consonants. Examine the values in the the sixth column, under the symbol 'e'. The values in this

	-	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
A	13	0	16	8	1	0	0	8	0	4	0	2	20	7	27	0	6	0	27	8
E	99	14	0	10	16	2	3	2	0	1	0	0	9	14	33	1	5	9	37	31
I	6	5	5	27	2	6	9	5	0	3	0	0	13	10	61	32	2	0	3	16
O	12	0	5	1	10	0	5	4	0	0	1	2	8	14	51	4	6	0	24	4
U	0	10	1	4	5	6	0	1	0	2	0	0	11	4	7	1	0	0	2	9
B	0	3	0	0	0	7	0	0	0	6	0	0	13	0	0	7	0	0	0	2
P	0	9	0	0	0	14	0	0	5	0	0	0	5	0	0	7	2	0	18	2
F	7	4	0	0	0	6	7	0	0	6	0	0	0	0	0	10	0	0	3	0
V	1	9	0	0	0	14	0	0	0	12	0	0	0	0	0	1	0	0	0	0
M	10	15	3	0	0	15	0	0	0	8	0	0	0	7	0	8	8	0	0	4

Table 2.

Vowels and Labial Consonants Compared

column are low for the vowel sounds, but high for the labial consonants. In constrast, columns fourteen and fifteen, under the symbols 'm' and 'n' have large values for all of the vowel sounds but much smaller values for the labial consonants. The similarity between the corresponding values in each set of five rows can be described as a correlation, and measured using familiar techniques, yielding a value for the similarity, interdependence, or interchangability of the two letters. The correlation between the rows in the first set reflects the fact that each row of data describes adjacency patterns for letters representing vowel sounds.

The correlation between adjacency patterns for vowel sounds has been noticed before, and used in an attempt at a definition of the difference between vowel and consonant (O'Connor and

Trim, 1953).

A more sophisticated statistical treatment can yield considerably more information about the sounds represented by the letters, as the scatter diagrams in the last section show.

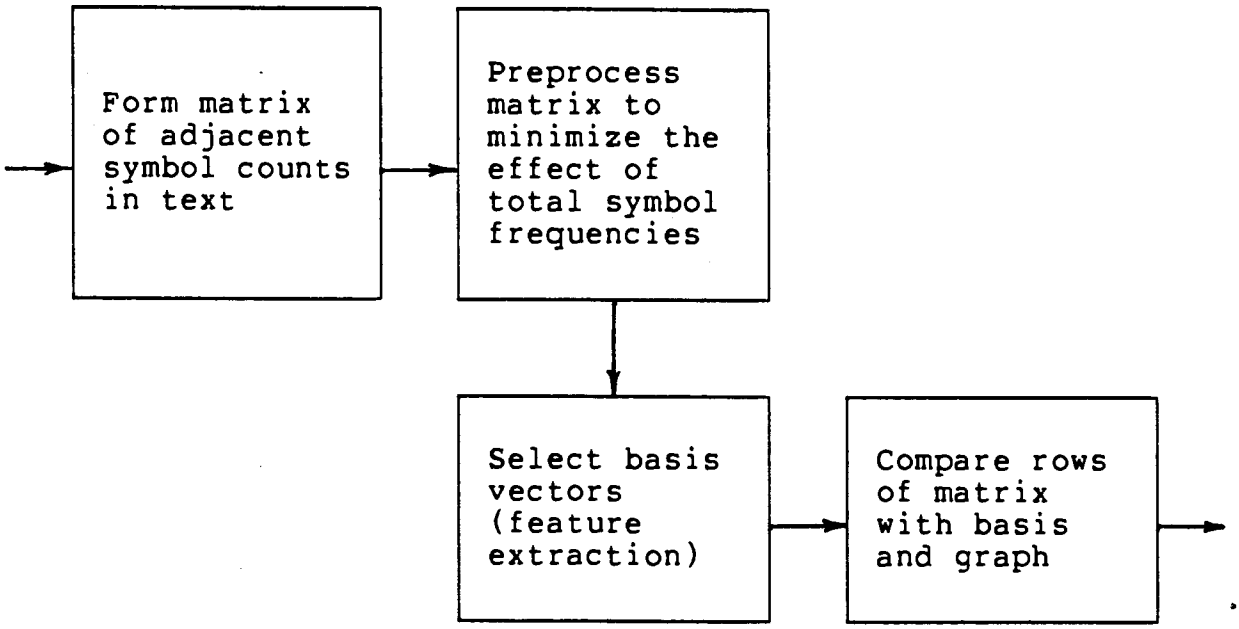


Figure 4. Method of Producing Scatter Diagrams

1.3 Computational Algorithms

This chapter discusses why and how the computational algorithms work, and gives further results.

To find suitable coordinate values for representing the letters of the alphabet, several distinct methods were employed. There are various arguments for and against each of these methods, and I will note some of these in describing the methods and presenting typical results. The results given below are remarkable because they show how an unknown symbol system may be

decoded, but the method used is not particularly remarkable. It is worth presenting an informal account of this method. In reading ordinary text a person often comes across a word that he has never seen before and whose morphology baffles him. If the word occurs several times in the text, the person may have a very good idea of what it means by the time he is through. How does he do this?

Introspectively, it seems that the meaning of an unknown word is found by looking at the context and imagining other, known words which might fit that context. It is rather as if each example of the word in a context is an equation, and the set of examples forms a system of simultaneous equations to be solved. If I say 'Put the blarque on the table' it is difficult to form any firm idea about the meaning of the word 'blarque'. If after that I say 'Take the cover off the blarque' and 'Help yourself to the rice in the blarque' and 'Watch out, the blarque is hot' then one can easily figure out what sort of thing a blarque is. It is not at all clear that one could write a computer program that would figure out that a blarque is a kind of cooking pot, but perhaps there is an easier task that could be programmed.

If a large sample of text, such as stack of books, is available, and the above sentences occur in that text, there is also an excellant possibility that more ordinary sentences such as 'Take the cover off the pot' may occur. Seeing this sentence with the analogous sentence above would suggest to anybody that

a blarque is a pot, or that a pot is a blarque. Without any real understanding of either word it is possible to see that the two words have a similar meaning. It is very easy to envision a computer program that would simply find out which words occur in similar contexts. If occurence in similar contexts implies a similarity of meaning, then some information about the meanings of words will have been obtained.

In the following section I show how this notion was pursued at a lower level, to find out the interpretation of alphabetical symbols.

Not all of the research performed can be described here, and much of the research that is described is given only a brief review. There are too many different combinations of possible data sources and methods: only a few can be presented in any detail. To illustrate the problem, here is a list of some of the choices that had to be made:

1. Languages to be studied
    a. related languages from one family
    b. single samples from different families
2. Type of sample used
    a. continuous text
    b. common phrases and typical sentences (assorted)
    c. lists of common words
3. Mathematical analysis
    a. Statistical methods
    b. Equation solving methods

c. Optimizational methods

4. Type of data collected

   a. Unsymmetrical data table

   b. Symmetrical data table

5. Amount of data collected

   a. adjacent symbols only

   b. nearby symbols at various distances.

6. Accuracy of data storage

After along period of attempting to evaluate these statistical methods for assigning coordinate values I have come to the conclusion that it is not necessary to choose the best method, since any one of a number of methods will produce a coordinate assignment that can be used by other programs. Some of the programs that can be applied, given an initial coordinate assignment, result in a more accurate choice of coordinate values. These latter programs are nonstatistical in nature and work by solving systems of equations. It is important to note that the information which is discovered by solving equations based on a text sample or utterance is essentially the same information arrived at with the statistical method, but there is more of it. We thus rediscover what should have been found by the statistical method. For example, the first purpose of the statistical method is to provide a set of coordinates for the letters and, later on at a higher level , a set of coordinates for words. Suppose one discovers with a statistical method a set of coordinates for the letters of the alphabet. Coordinating the

text and treating it as equations, then solving them, results in a series of coordinates. While it may also be other things, this series will be another set of coordinated values, and a more accurate set, since it is based on a nonstatistical method.

Such accurate sets of coordinates, in which data compressions would be better and smaller text samples used, could then replace the original sets of coordinates. Alternately, the same text sample could be used at a higher level of accuracy. Such considerations suggest a boot-strapping method by which an initial approximate set of coordinates would be revised, creating better coordinates which could, in turn, produce yet better.

Statistical studies of language have usually been more concerned with obtaining representative samples for analysis than they have been with grammaticality or acceptability, since the addition of a grammatically incorrect sentence to a sample may not affect the results of the study at all, whereas the censorship implicit in obtaining a sample that is perfectly correct may be enough to make the sample nonrepresentative. Transformational grammarians, on the other hand have focussed on obtaining purely grammatical sentences for analysis, even if such sentences do not reflect actual usage.

The methods discussed above suggest a new view of deviant usage. Modern linguistics has often claimed that idiosyncratic usage may be perfectly valid and correct personal language; on this view such a language is impossible, since such a language

would allow the possiblity of genuinely deviant text. We could define deviant text as text whose statistically derived coordinates are different than those derived by an application of the boot-strapping process. Ordinary text derived from casual conversation would tend to be nondeviant; the statistical properties of it would tend to resemble properties discovered by the complete analysis. The results of such a study could include a common or central set of coordinate values that could be considered absolute or universal.

1.3.1 Factor Analysis of Symbol Counts

A table of adjacent symbol counts can be treated in two different ways:

1. as a set of 27 vectors
2. as a matrix describing a linear transformation.

Both of these approaches lead in the same direction, and the distinction between them is similar to the distinction between statistical and nonstatistical methods discussed previously. In treating the matrix of adjacent symbol counts as 27 pieces of data about each of 27 items, statistical methods were employed to condition and describe this data. Alternatively, the matrix of adjacent symbol counts can be used as a transform applied to some initial estimate of the solution to a system of equations, and this method is essentially nonstatistical. The first successful treatment of the data actually involved both methods; this will be explained after

each approach is described separately.

The most familiar use of the data treats it as a collection of measured values for each of 27 distinct objects. A description of the statistical techniques applied to the data will be supplemented by an explanation of its origin and significance in terms from associationist psychology.

An optional preliminary method for factor analysis is the preprocessing of the data. This ought to at least include subtraction of the means of each column of corresponding coordinates. A preliminary examination of the adjacent symbol count data suggested that this was not enough preprocessing, and that for the results of factor analysis to be significant other steps would have to be applied. One reason for this is that certain row vectors representing counts of symbols adjacent to rare letters such as 'q' and 'z' are almost entirely filled with zeros. If the means of each column are subtracted, all of the vectors representing rare letters will become more similar, and then these letters will be placed in similar places on the scatter diagram. (see Figure 5.)

This is not an error, but merely an unwanted result. The simple preprocessing method of subtracting column means emphasizes a factor that is indeed a significant difference between letters: frequency of occurrence. For some purposes this factor might be considered important, I view it as resulting from historical accidents in the development of writing, and want to minimize its importance.

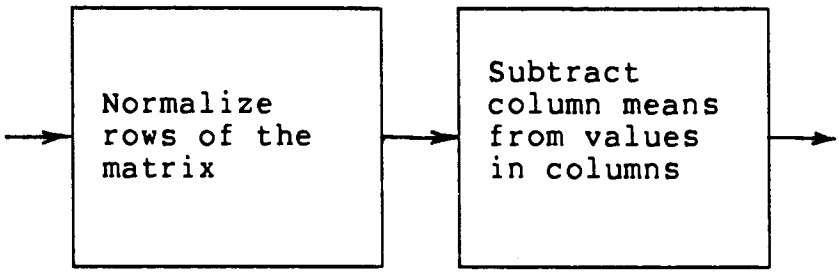


Figure 5. Simple Preprocessing Method

To emphasize factors relating to the sound of letters and de-emphasize absolute letter frequency, row vectors can be normalized before column means are calculated and subtracted. This combination of preprocessing operations was the minimum that produced acceptable diagrams in which the positions of the letters reflected their sounds.

An improved set of preprocessing operations replaced the two basic operations of normalizing rows and subtracting column means with related operations. (see Figure 6.) These operations are the simplest versions of two more general types of operation, which involve subtracting expected values and weighting by an estimate of significance.

The mean of each row or column is the value that would be given if one were asked to guess a particular value, told only that it occurs in a certain row or column. Subtracting such an expected value is a typical operation in pattern recognition, where one wants to maximize the impact of each datum by

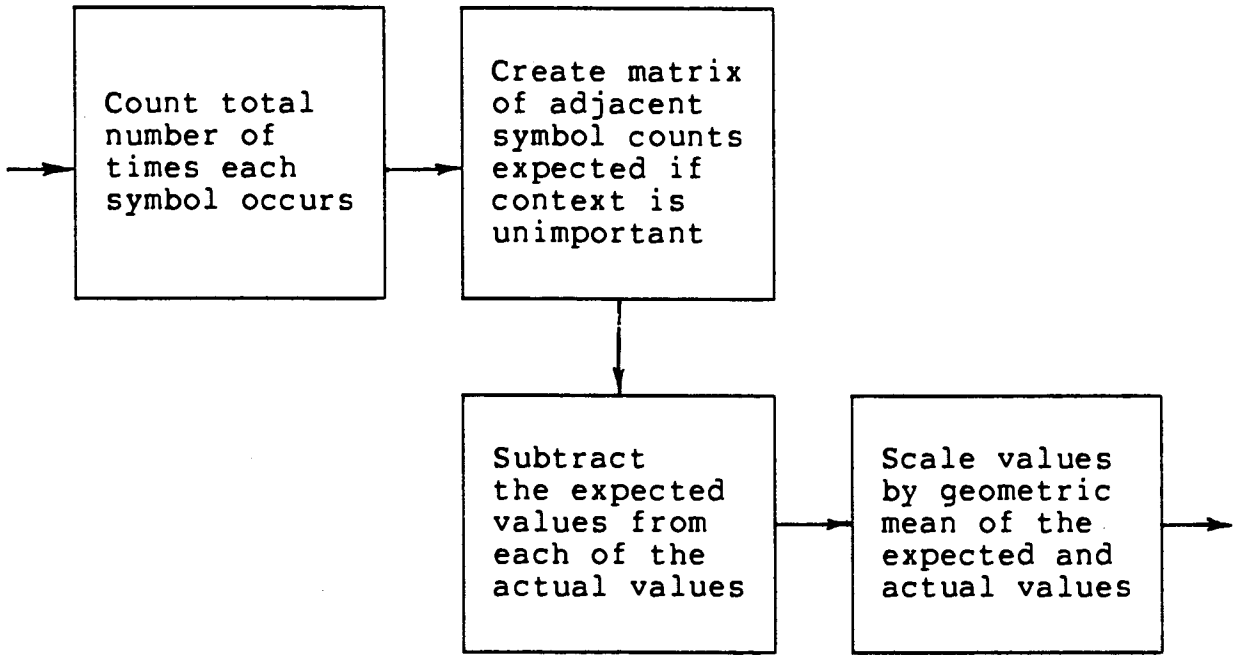


Figure 6. Better Preprocessing Method

comparing it with the best estimate that can be made before the actual data is available. The mean of each column in the matrix of counts is proportional to the number of occurences of that letter. Similarly for the mean of each row. The best guess we can make for a particular value in the matrix is not either of these, but is proportional to their product, and represents the probability of one of the letters occuring before the other letter. This probability can be calculated by knowing how frequently each letter occurs in the text. Thus, better expected value is derived from the product of row and column means. Thus, the expected number of time 's' appears before or after 't' is

the product of the total number of times 't' appears and the total number of times 's' appears, divided by a constant. This expected value could be described as the number of times 's' would occur with 't' if the letters in the text sample were shuffled randomly. In other words, the expected value is based on the number of times each letter occurs but not on the syntactic properties that letters have. Subtracting the expected value, is an operation on individual values in the table, not on rows or columns.

A similar version of the normalizing operation can be made by producing an 'expected scale' value. Rather than normalizing the rows or columns, the differences between calculated and counted values can be scaled by the mean of these two values. The result is a table of variances, showing how the actual counts of adjacent symbols compare with the number of adjacent symbols that would have been found if the same number of symbols had been distributed randomly in the text. Scatter diagrams based on this series of operations are similar to ones based on simply subtracting means and normalizing, but the positions of the letters seems to match my linguistic intuitions better. One such diagram appears in the next section, and several others obtained from texts in various languages are discussed in a section devoted to linguistic analysis of results.

Various combinations of the ordinary and revised versions of the two basic preprocessing operations were tried, with a variety of results. For most of the samples used, the two

revised operations in the order described seemed to produce the best results upon subsequent factor analysis, but this cannot be considered a proof of their optimality. Two forms of factor analysis used are shown in figures 7 and 8 on page 61.

An alternative to either form of normalization was also used in some experiments. From a consideration of the Law of Frequency in associationist psychology, the logarithm of each value in the table was taken. This also led to improved results in some experiments.

In the next section I will present results based on a single one of the various combinations of methods of preprocessing mentioned here. The lengthy section of this research that involved attempting to find the best preprocessing method, and included other attempts at finding a good statistical approach to the data, was beset with difficulties similar to those that trouble social scientists and others attempting to apply mathematics in the absence of a thorough theoretical understanding of the problem they are studying. The class of nonstatistical methods to be discussed next is simpler, since it involves framing specific theoretical requirments.

1.3.1.1 Representation of Results

The results of the above method may be represented as scatter diagrams, in which the letters of the alphabet are displayed at various positions on the page according to the values of two coordinates. Figure 9 gives a typical result for

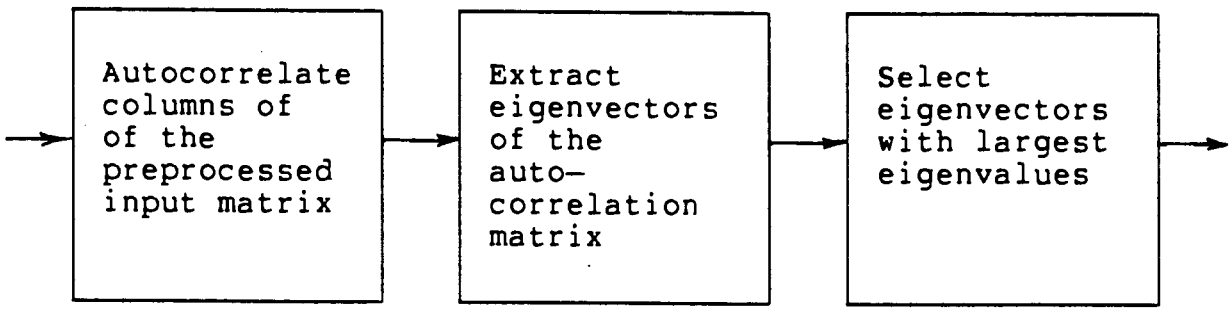


Figure 7. Simple Form of Factor Analysis

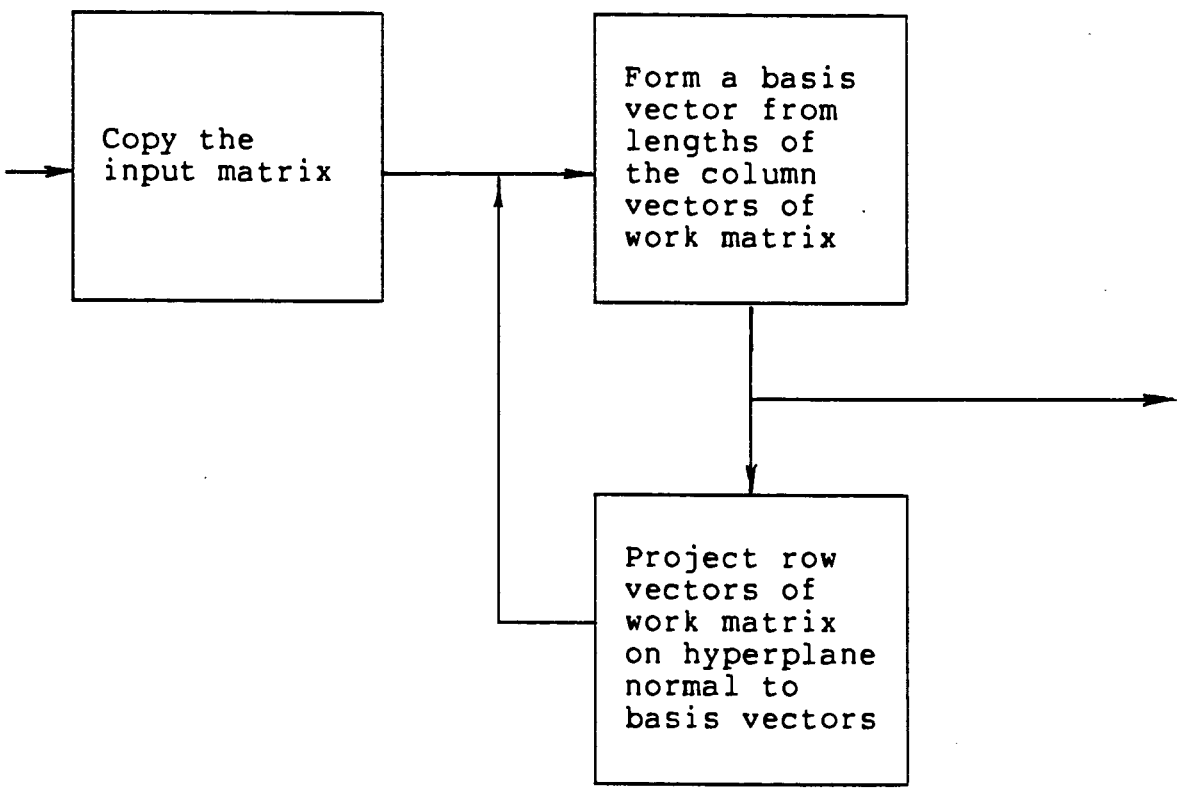


Figure 8. Modified Method of Factor Analysis

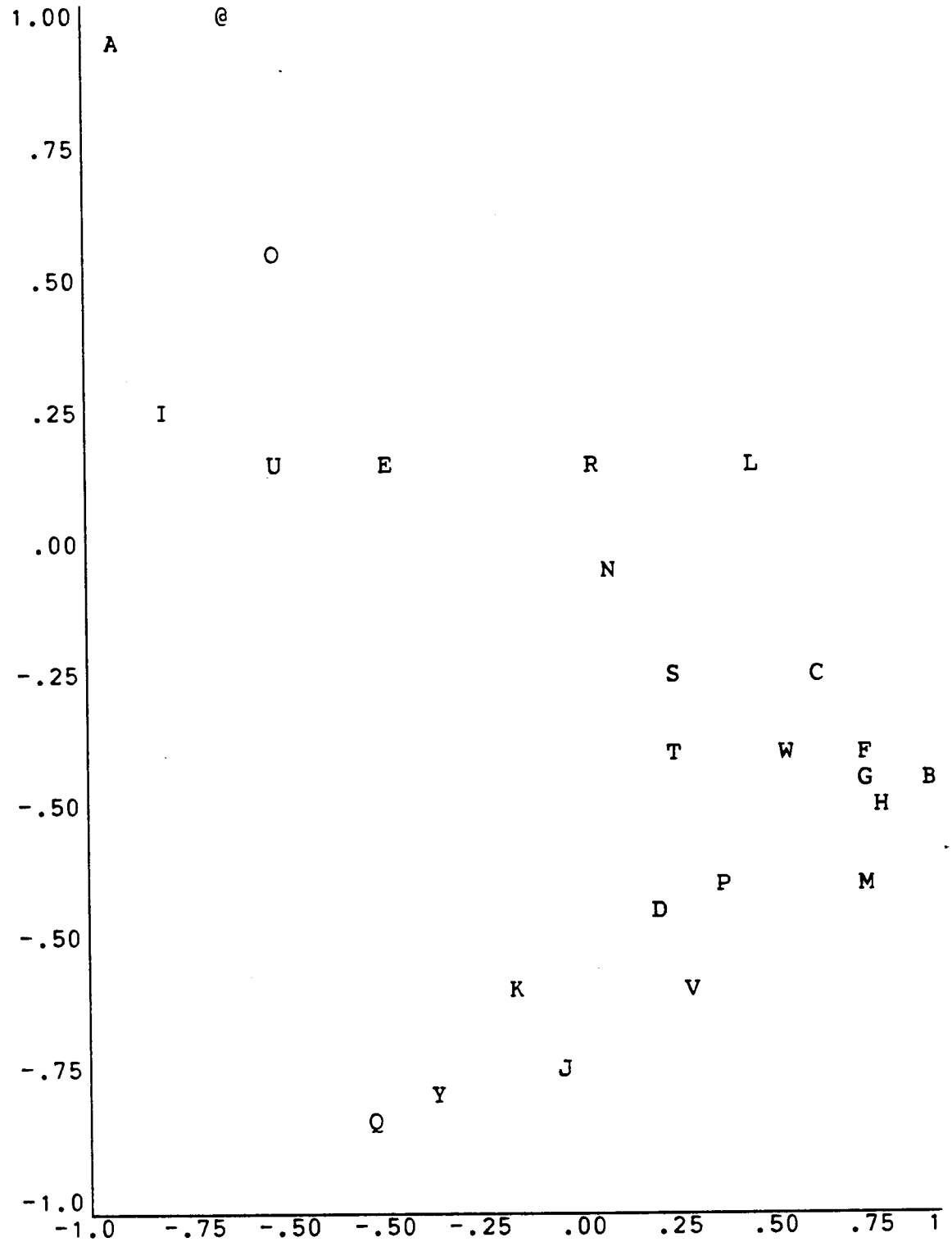


Figure 9. Analysis of English Word List

English text.[9]

This diagram is far from perfect. It is important to realize that the best two sets of coordinate values according to the criteria employed in factor analysis may be only marginally better than several other sets. Other sets of coordinates may reveal that certain letters shown as neighbours on this diagram are actually quite distant in the higher dimensional space that is projected onto the two-dimensional diagram. There is no avoiding the fact that the symbols are being represented by points in a space of many dimensions, and that no two dimensional scatter diagram can properly convey all of these.

It can be shown through the use of these coordinates in programs such as described later that they are acceptable as a representation of the sounds of the letters. To some extent this can be seen by merely examining the scatter diagrams, and noting that letters of similar sound tend to occur nearby. With this particular method the number of coordinates required for a good representation of the data is quite high, and so individual scatter diagrams tend to be confusing and hard to evaluate.

For example in Figure 9, above, the cluster in the lower right hand side contains all labial consonants, except for the letters 'G' and 'H' which seem out of place. Actually their position makes sense when you consider the number of English words containing 'ough' in which the 'gh' sequence is pronounced

---

[9] This diagram is reproduced again in the last chapter, where it is analysed in attempt to determine its linguistic significance. Similar diagrams for other languages are also discussed.

as if it was 'f'.

1.3.2 Optimization Method

All of the methods described here are closely related to those of a branch of statistics known as multivariate analysis. In multivariate analysis it is assumed that the observations can be described in terms of a function of more than one variable: that there are some unknown variables which can vary independently of one another, and that the observations to be accounted for depend in some way on all of these variables together. A typical problem is then to recover the underlying variables and describe the effect of each of them on the observables. Notice that the underlying variables are themselves often not directly observables; they constitute a level of underlying structure analogous to the deep structures of transformational grammar.

Printing text makes use of twenty six alphabetic characters plus the blank space between words and several punctuation marks. The number of possible combinations of these symbols into two and three letter strings depends on the number of symbols used, but is quite large. However, only some of the possible strings of symbols occur. Roughly 70% of the two-letter digrams do actually occur in a lengthy sample of text, although most of these are quite rare. Why does a common trigram like "and" occur more often than an uncommon one like "jxl"? Obviously it has something to do with the fact that the latter is almost

impossible to pronounce. If the transition between "a", "n" and "d" in and are smoother and more easily managed than those between "j", "x" and "l", then one should be able to express this fact mathematically. If one uses the letters to stand for their representation as vectors, then one could try to express this by saying that

$(n-a)+(d-n) < (x-j)+(l-x)$

This quasi-mathematical representation uses (n-d) to stand for the difference between "n" and "d" or, perhaps, for the amount of effort it takes to move the speech organs so that they will pronouce a "d" after an "n". There are a number of different mathematical statements which could be used to state this observed inequality more precisely. However it is expressed, it should be clear that it is just one of a great many inequalities that could be formulated, asserting that sequences of sounds which do occur are made with less effort than those which do not.

Note that since "and" occurs very often, there is presumably some simple and natural transiton from "a" to "d" that passes through "n". The most obvious example of such a transition would be one in which "n" was exactly halfway between "a" and "d". This could be expressed by

$n = a/2 + d/2$

However, the mere fact that "and" is a common string of symbols does not guarantee that this holds. We may assume that "n" is only approximately between "a" and "d". This can be

expressed by saying that

$$n = (a2 + d/2) - X$$

where X represents the unknown discrepancy between "n" and the point halfway between "a" and "d".

Variables such as "X" above are called "slack variables". They are introduced to "take up the slack" between the approximation and the ideal. Each occuring trigram can easily be converted to an equation involving a slack variable. However, each such equation must have a different slack variable, so that the number of variables is always more than the number of equations, no matter how many equations are constructed. This means there will be no true solution to the problem, and we have to find a best approximate solution by minimizing or maximizing some combination of variables.

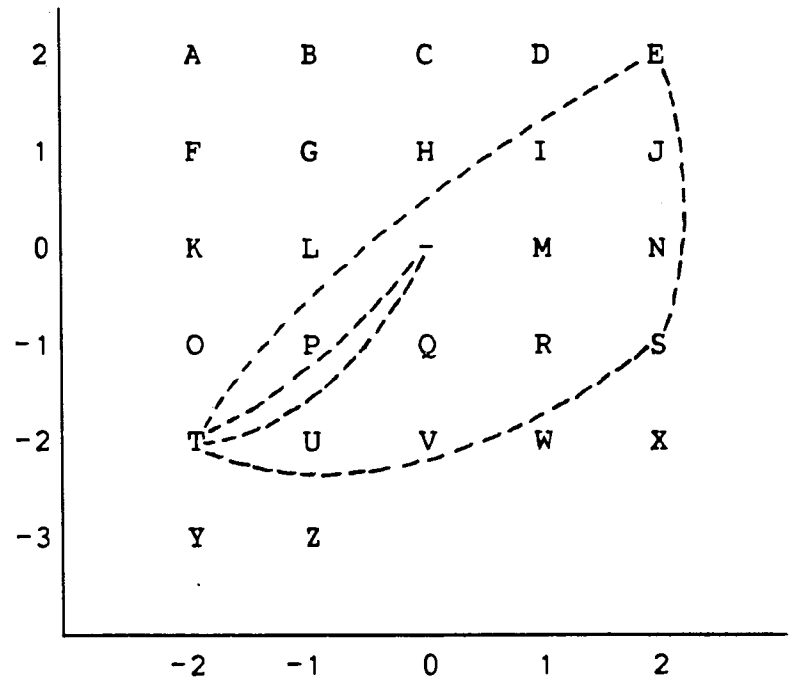
In treating the assignment of coordinate values as an optimization problem, it is necessary to find some numerical value representing the "goodness of fit" or appropriateness of a particular assignment. One such measure is the sum of all of the differences between the values assigned to successive letters. This is just the sum of the slack variables, and my minimizing it we minimize the 'slack', or reduce the length of the path traced by the text sample in some multidimensional vector space.

If we take the length just described as a quantity to be minimized, we can use a simple optimization algorithm to assign coordinate values, or to choose amongst possible assignments. This has advantages over the method discussed previously. With

the earlier method values are assigned to two letters if their statistical properties are similar. This is certainly a desirable feature for some purposes, since it reflects a genuine property of the set of symbols. But for algorithms such as those to be discussed below, it is a source of difficulty. These algorithms make use of the inverse process of finding a letter to match a set of values, and are easily confused if two letters have very similar values. This problem will be discussed further in the next section.

Figure 10 shows the basic principle employed in optimizing a coordinate assignment. The initial assignment is based on the alphabetical ordering of letters, while the final one is obtained by applying optimization to minimize the length of the trajectory representing a text sample.

The example given below shows all of the steps of an optimization process used to find two coordinate values for each of the symbols used in a selection of Spanish text. Each rectangular array of letters is organized to reflect the values currently assigned, using the same coordinate system as in Figure 8. The hyphen near the center of the diagram represents the space between words. In the initial assignment the letter 'a' appears in the upper left hand corner because it is temporarily assigned the coordinates -2,-3. The letter 'b' is assigned -1,-3 and the letter 'f' below the 'a' is assigned -2,-2 while the space between words is assigned 0,0 . With this assignment of values the total length of the function



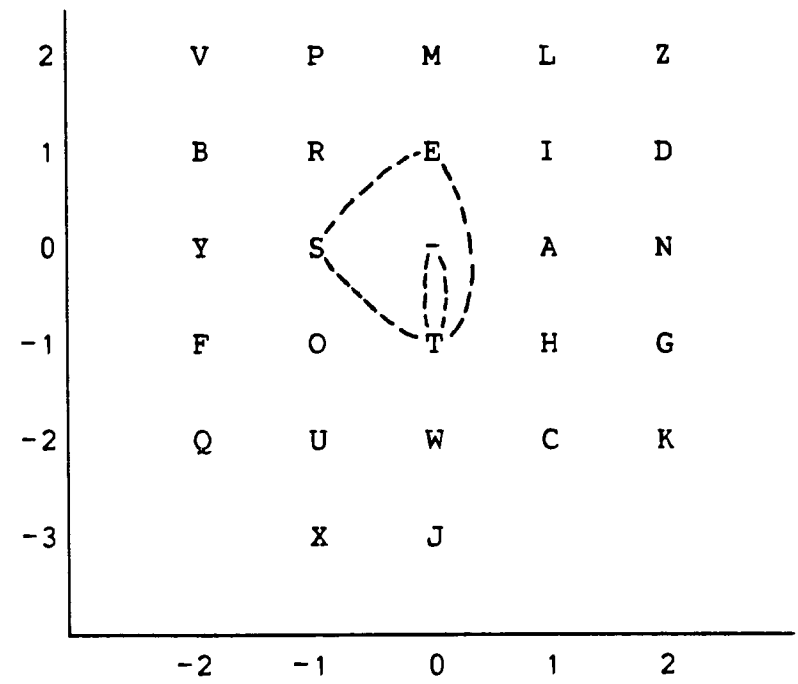
Initial Coordinate Assignment

	-	T	E	S	T	-
x:	0	-2	2	2	-2	0
y:	0	2	-2	1	2	0
d:		4	8	3	5	4

Total distance

= 24

------ indicates trajectory metric: $d = |x - x| + |y - y|$



Final Coordinate Assignment

	-	T	E	S	T	-
x:	0	0	0	-1	0	0
y:	0	1	-1	0	1	0
d:		1	2	2	1	1

Total distance

= 7

Figure 10. Optimization Example

corresponding to the text sample is 14412. (see Table 3.)

Each successive array of letters shows a slight alteration in the coordinate assignments made by swapping the values assigned to two letters. For illustrative purposes the swapping is chosen by exhaustive examination to yield the greatest possible improvement (reduction in total length). Practical use of this method can involve more sophisticated methods of choosing pairs of letters to swap.

For example, the second array of letters differs from the first only in the exchange of 'e' and 'm', so that 'e' is assigned the coordinates 2,0 instead of 2,-2. The final array represents an assignment of coordinates that gives the whole function a total length of 8732. No further improvement is possible by swapping of single letter pairs.

The resulting coordinate assignment seems to reflect the phonology of Spanish as well as might be expected for a simple two dimensional solution. All of the vowels are clustered around the center of the diagram. There are not enough vowels in the language, so the same inner square of letters includes the continuents 'n' and 'l'. The letter 'c' is also in this inner square, but this must be an anomaly. (The position of 'c' in the English example below is more appropriate.) We may also notice that 'y' and 'l', which are often the same sound in Spanish, are neighbours, and that the upper right hand corner of the diagram contains the letters 'b', 'm', 'p', and 'v' representing labial sounds.

Initial Length is 14414

```
a b c d e      a b c d m      h b c d m      h b c d m
f g h i j      f g h i j      f g a i j      f g a i j
k l - m n      k l - e n      k l - e n      k l - e n
o p q r s      o p q r s      o p q r s      q p o r s
t u v w x      t u v w x      t u v w x      t u v w x
y z . . .      y z . . .      y z . . .      y z . . .

h b c d m      h b c d m      h b c d m      h b c d g
f g a i j      f g a i j      f g a i j      f m a i j
y l - e n      y l - e n      y l - e n      y l - e n
q p o r s      q p o r s      q p r o v      q p o r v
t u v w x      t u v w x      w u s t x      w u s t x
k z . . .      k z . . .      k z . . .      k z . . .

h b c r g      h b c r g      h b c r g      h b c r v
f m a i j      f m a i j      f m a i j      f m a i j
y l - e n      y l - e d      y l - e d      y l - e d
q p o d v      q p o n v      q u o n v      q u o n g
w u s t x      w u s t x      w p s t x      w p s t x
k z . . .      k z . . .      k z . . .      k z . . .

h b c r v      h b c r v      h b c r v      z b c r v
f m a i p      z m a i p      z m a i p      h m a i p
y l - e d      y l - e d      y l - e d      y l - e d
q u o n g      q u o n g      q u o n g      q u o n g
w j s t x      w j s t x      w j s t x      w j s t x
k z . . .      k f . . .      k . f . .      k . f . .

z b m r v      z b m p v      z b m p v
h c a i p      h c a i r      h c a i r
y l - e d      y l - e d      y l - e d
q u o n g      q u o n g      q u o n g
w j s t x      w j s t x      w f s t x
k . f . .      k . f . .      k . j . .
```

Final Length is 8732

No further improvement is possible by swapping of single letter

For comparison purposes the final diagram for Spanish is repeated again below beside the diagram obtained by applying the same program to a sample of English text.

Spanish	English
z b m p v	v p m l z
h c a i r	b r e i d
y l - e d	y s - a n
q u o n g	f o t h g
w f s t x	q u w c k
k . j . .	. x j . .

The same algorithm can be applied to a three-dimensional pattern of letters. Typical results for Spanish and English are given below. For each language three small square patterns of letters are given. They represent successive layers of a three-dimensional structure and should be imagined as lying one above the other.

Spanish			English		
h c b	a l m	z g k	z m k	i d n	g f o
i d r	e - s	o y p	v w p	a - e	r t h
v t j	u n q	x f w	q b x	y s l	j c u

1.3.3 Filtering

The account of linear Markov filters given previously suggested that they could be considered as equivalent to the hypothetical device which translates deep logical form into a recognizable human language. If this is true, it would be an interesting project to create a linear Markov filter of high order for each of two languages, invert one of them, and then attempt machine translation by using the inverted filter to remove the correlations of one language and the other filter to impose new correlations on the result. Unfortunately, even a filter of order 10 would present extreme computational difficulties because of storage requirements. An order 10 filter would use vectors of length 270, and these would be transformed by matrices containing a total of 72,900 values.

The reduced set of coordinates provided by factor analysis seems an obvious way of overcoming this difficulty. If the two coordinates assigned to letters in creating a scatter diagram such as the one shown above were employed, then a filter of order 10 would only be a 20 by 20 matrix, and this could easily be handled on a small computer. The appropriate matrix would be created by a process called autocorrelation, described in detail in another section.

In the course of this research various autocorrelation matrices were constructed and tested as filters, both with and without feedback. Small autocorrelation matrices were used successfully in simple filters that acted as low-order Markov

filters, but higher order filters of this form were not satisfactory. In fact, higher order filters made from autocorrelation matrices behaved just like lower order filters.

This result was disappointing and puzzling. It was difficult to find any reason for the failure of high-order autocorrelation matrices to perform better than lower order ones. The application of factor analysis seemed successful, since the results closely approximate our understanding of the relationships between sounds represented by the letters, yet the autocorrelation matrices could not be made to work. The reason for this became apparent on inspection of the actual values of matrix elements. Elements of the matrix representing correlations between letters more than three spaces apart in the text were almost zero, and became closer to zero in larger text samples, as random fluctuations cancelled out. A possible reason for this did not appear until a different model of grammar was studied.

Since it only seemed possible to create low order filters using the coordinate values found through factor analysis, it seemed worthwhile to investigate low order filters. One possible use of these filters is in simulating the process of assimilation by which most phonetic changes in the historical development of languages are known to occur. None of my attempts at simulating phonetic change through assimilation were very successful, but in the process of applying simple filters to various text samples, it was possible to observe an interesting

property of common phonetic sequences. In analysing this, new ideas about the nature of grammar were developed, which explained the difficulties encountered in all attempts at filtering text. A few examples of filtering will suffice to demonstrate these problems and the effect that was observed.

1.3.4 Filtering Examples

To avoid problems with varying word lengths, a set of words was chosen that had the same length in each of five languages:

Latin	French	Spanish	Italian	Portuguese
contra	contre	contra	contro	contra
liber	livre	libro	libro	livro
catena	chaine	cadena	catena	cadeia
octo	huit	ocho	otto	oito
lingua	langue	lengua	lingua	lingua

The following two examples are with a small amount of filtering. The coordinate values used were from the optimization method.These example are based on a simple two-dimensional model (two coordinates assigned to each symbol).

The numerical values given are filter coefficients. Each coefficient represents the amount of influence a particular coordinate of neigbouring symbols has on the values of the current symbol. Two filter coefficients are given since in each of these example two coordinates per letter were used, and the amount of filtering in each dimension were listed seperately.

The first example has twice as much filtering on the first as on the second coordinates.

Filter coefficients 0.1 and 0.05:

Latin	French	Spanish	Italian	Portuguese
contra	contre	contra	*contdo	contra
liber	livre	libro	libro	livro
catena	chaine	cadena	catena	cadeia
*olto	*huet	*ocho	otto	*oeto
lingua	langue	lengua	lingua	lingua

Notice that in this example there are few changes. Italian 'contro' became 'contdo' but there were no changes in the equivalent words of other languages. On the other hand, the Latin word 'octo' and its derivates in French and Portuguese changed, while the Italian and Spanish equivalents remained the same.

The next example is with more filtering on the second coordinates.

Filter coefficients 0.05 and 0.1:

Latin	French	Spanish	Italian	Portuguese
contra	contre	contra	contro	contra
liber	livre	libro	libro	livro
catena	chaine	cadena	catena	cadeia
octo	*cuit	ocho	otto	oito
lingua	langue	lengua	lingua	lingua

Notice that although the total amount of filtering is the same as in the last example, the only change is the French 'huit' becoming 'cuit'.

One more example can be given to show more drastic filtering. In the following case just twice as much filtering was used as in the preceding example.

Filter coefficients 0.1 and 0.2:

Latin	French	Spanish	Italian	Portuguese
*aontia	*aontie	*aontia	*aontdo	*aontia
*lam-i	*lapre	*labio	*labio	*lapro
catena	*chcine	*caeena	catena	*caeeia
*o-to	*cuet	*occo	otto	*oeto
*lingoa	*lango-	*lengoa	*lingoa	*lingoa

In this example I have used the hyphen '-' to indicate the character normally used for the space between words. This might be considered as the vanishing of the original letter. Thus Latin 'octo' is written above as 'o-to' but might better be written 'oto' or even 'otto' where the duplicated letter 't' might not be pronounced as parts of separate syllables. This latter form is identical to the Italian 'otto' which must be noted as one of the stablest forms of those shown.

The existence of stable forms is interesting and leads to many questions. Why are the Italian and Latin words 'catena' and

the Italian word 'otto' the only forms that undergo no changes with this much filtering? What difference between 'liber' and 'catena' explains why the former is changed almost beyond recognition into 'lam-i' while the latter is unchanged?

1.3.5 Invariants and Grammatical Theory

Forms that are unchanged by a transformation such as the above filtering operation are called invariants. There is a different notion of grammar that can be developed using the notion of invariants. The linear Markov filters mentioned earlier are interesting as they relate to one theory of grammar that proposes transformational processes acting on some common core of all human languages to impose the properties of a specific language. Another view of grammar has focussed on grammars as acceptors or correctors of 'improper' usage. A filter could be imagined that would mark unacceptable forms in some way, or even correct them. Suppose a grammatical filter of this sort is given ordinary correct text as input. What should it do? Clearly, it should just pass the text through without any change at all. Grammatical text samples should be invariant under the filtering operation. To investigate such grammars involved preparing filters that would pass ordinary English through unchanged but alter any 'unrecognized' forms.

In matrix algebra we do not always have true invariants which remain unchanged by transformations, but we can usually define certain vectors which are invariant with respect to

orientation. Such vectors, called 'eigenvectors' or 'characteristic vectors', are only changed in magnitude by a particular linear transformation, and can serve as a part of a definition of that transformation. It is possible to design a linear transformation for which the eigenvectors are true invariants: unchanged either in magnitude or direction. If all of the eigenvectors are true invariants, then the transformation is just the identity. A more interesting and useful transformation is one for which some of the eigenvectors are true invariants and the others are anihilated (changed to zero vectors). I refer to such a transformation as a pseudoidentity transformation, to emphasize its similarity to the identity.[10] For many inputs it behaves exactly like the identity, while for others it performs radical changes.

A pseudoidentity filter must be constructed using enough of the eigenvectors to capture the properties of the text. The version constructed for English began with vectors of length twelve, so the autocorrelation matrix had twelve eigenvectors. Use of all twelve leads to a genuine identity filter which passes the origal text unchanged:

THE SENTENCES HERE ARE FOR USE AS AN EXAMPLE OF ENGLISH TEXT

If only the six largest eigenvectors were used, the resulting

---

[10] In mathematical terms this particular pseudoidentity transformation is just a projection, but I will argue that this is not always the case.

pseudoidentity filter passed the text with many errors:

THT SENTERCES HTRT ART BOR USE AS AN ERAUUDE OB TNBDUSH EERT

With the use of nine eigenvectors the original text was passed correctly. (See Figure 11, page 80)

Notice in this example that some of the words made up only of very common letter sequences were passed correctly by the pseudoidentity filter based on six eigenvectors, while a word like 'ENGLISH' which include the unusual sequence 'NGL' was considerably distorted. In general as more and more eigenvectors are used the filters constructed pass more and more words correctly, beginning with those containing the most common sequences of letters. By the time nine eigenvectors are used, all of the sequences in typical English text are accepted, but some unusual sequences will still be altered. With the final addition of the twelfth eigenvector the filter becomes a real identity matrix and accepts any sequence of letters.

The filter created by using nine eigenvectors is adequate as an example but is very limited in its abilities. This type of filter could in principle be of any order, and we may assume that as with linear Markov filters, the higher-order examples would be better. In a later section I suggest that the class of linear Markov filters and the class of pseudoidentity filters are not disjoint: it is possible to construct a filter that can play both roles.

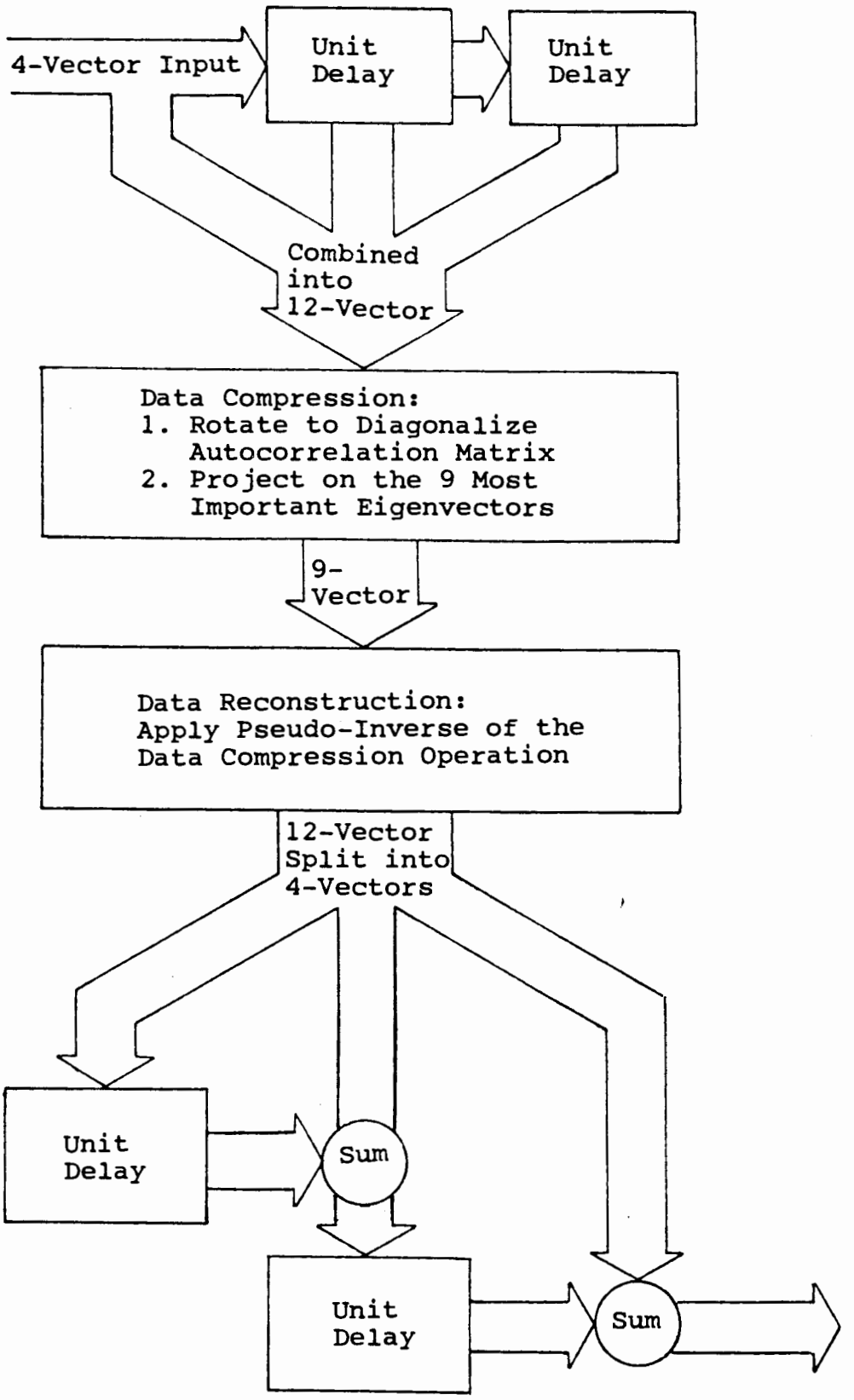


Figure 11. Pseudo-Identity Filter

Of particular interest is an unexpected property of some pseudoidentity filters: they can act as associative memories. This will be explored in a later section.

One problem with the pseudoidentity filter as described is that it involves a loss of information, which certainly seems an unlikely property for any memory-like device. Technically this kind of filter is a projection, which maps a multidimensional space onto a subspace, and in doing so information is lost. It is difficult at first to see how the grammar for a language could possibly be a projection. Surely it should be possible to encode whatever information is to be transmitted without any necessary loss. Indeed, it is usally supposed that human language is highly redundant, with information being encoded to ensure correct transmission.

In the section on associative memory I note that feedback can be added to a linear corrective or pseudoidentity filter to change it from a projection into a nonsingular transformation that passes all information without loss, while retaining the desirable properties that make it a model for human language processing.

1.3.6 Markov Filters

The usual way of implementing such a filter begins with an analysis of a text sample in a natural language. For a first-order Markov filter we need to know the probabilites of each symbol following each other symbol, and this can be found

by counting the number of times each symbol actually does follow each other in a large enough text sample. The result of counting adjacent symbols is a matrix such as given earlier. Each row of this matrix represents a letter of the alphabet, in the usual order, and the numbers in that row represent the number of times each other letter occurred immediately following the given one, and thus the probability of its occurrence in that context.

To use this matrix as a filter we must first find a representation of the letters of the alphabet as vectors. One such representation is interesting because it allows the matrix of adjacent symbol counts to be used with only minimal modification. Let each vector have as many components as there are symbols in the alphabet. For each symbol, define a vector that consists of zeros for all components but one. The component whose position in the vector corresponds to the position of the letter in the alphabet should have the value one. Thus the letter 'A' is represented as

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

while the letters 'B' and 'C' are

0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

and

0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ,

respectively.[11]

If any of these vectors are multiplied by the matrix of adjacent symbol counts in the usual way, the result is a vector identical to one of the corresponding row of the table. Although the conclusion of this process is obvious for such a simple example, we should follow it through to its conclusion since subsequent examples will use the same approach. The output vector is not exactly the same as any one of the input vectors, so we must find the best match. The usual way to do this is to take the inner product (or 'dot product') of the output vector with each of the vectors previously assigned to letters of the alphabet. Since each of these vectors has the value 'one' in exactly one place, and 'zero' everywhere else, the inner product can be obtained by inspection. If the output vector is

9 5 0 5 3 5 9 0 0 5 3 7 9 2 5 8 2 5 4 8 5 4 2 6 8 6,

and we are comparing it with the vector for 'A',

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 , then the inner product of these two vectors is just the number 9 in the

---

[11]To take less room on the page these vectors and those on subsequent pages have been written as rows of numbers, even though they may be column vectors. An alphabet of twenty-seven symbols is assumed here, counting the space between words as a symbol and ignoring punctuation.

first column of the output vector. The best match to the output vector corresponds to the largest value of the inner product, and that will be just the largest component of the output vector.

Suppose we wanted to find the most probable output following a given output. The simple filter just described gives the most probable output for a given input, so we need only turn the last output into an input by adding a feedback loop. For a Markov device we don't want to always get the most probable outputs, instead we want the more probable outputs to occur more frequently. If the only input was the previous output, feedback, the device would be completely determinate, and so a source of random vectors must be added to the input. If this is done with appropriate amounts of feedback and added random noise input, the result can be identical to an ordinary first-order Markov device. Indeed, the actual implementation of any first-order Markov device involves the same operations, although they are not described in terms of vectors, filtering, or feedback. The diagrams on the following page illustrate these steps.

The same operations can be performed with vectors having twice as many components and representing pairs of letters. The result is a second-order Markov filter. Extension to still higher orders is limited only by computer processing ability and memory, in exactly the same way as ordinary Markov generators are limited. In general the higher the order of the device, the greater is the similarity between its output and genuine samples

of whatever language was used as a source of data. It should be noted that the filters described above are linear devices, since matrix multiplication is a linear operation and feedback applied to a truly linear device does not make it nonlinear.

What is the relationship between Markov filters as just described and the hypothetical device for translating expressions in some underlying logical form into natural language? I suggest that they are the same.

1.3.7 Examples Based on Markov Filters

The algorithm used in creating the pseudoidentity filter described above requires extraction of eigenvectors as a data compression scheme. This is difficult to do for larger matrices, and limits the order of the filter. An entirely different algorithm can be employed, reducing the memory requirements but increasing processing time. This algorithm is much closer to those used in creating Markov devices: it stores sections of text and their probabilities, or a continuous length of text in which sections are repeated as often as necessary to encode their probability. Indeed, since the probability of a short section of text occurring is estimated by measuring the number of times it occurs in a given sample, a large sample of text may be stored without modification, and used to calculate the necessary probabilities as they are needed.

This algorithm was used to construct Markov filters that could also be described as pseudoidentity filters and as

corrective filters.

In implementing this algorithm, a short sample to be filtered must be coded as a vector, and compared with vectors representing sections of text in the larger sample that is stored as a source of data. A machine language routine was used to quickly encode sections of text, so that it was not necessary to encode the whole internal data sample before running the program. One routine quickly found the inner product of the two vectors representing the sample and successive sections of the internal text. The result of this comparison was a text selection of text representing the best match between samples. This text section was not output in its entirety, since this would limit high-order filters to whole sections of the internal text. Instead a single letter from the center of the matched text was output. As with the pseudoidentity filter described before, the program took single letters as input and output single letters, but a delay was used internally to store letters until strings of the appropriate order were collected. Several versions of this program were studied, both with and without feedback. The best of these was used to produce the sample given below. This program implemented a filter of order $2n+1$, using the last n values of the output, juxtaposed with the next $n+1$ values of the input to produce a single string of length $2n+1$. The device incorporated a delay of n spaces, so the juxtaposition of parts of output and input produced a string resembling a continuous segment of the input. This string was

used in the routine described above, and the best match found. As before, only the middle letter of this matching string was used as output.

The examples below illustrate the three aspects of this filter: as Markov device, as pseudoidentity filter, and as corrective filter.

The filter used in this algorithm uses an internally stored text sample, and the quality of the output depends on the size of the sample. In the examples given above the stored text sample was the first chapter of the book of Mark in the New Testament, and the output visibly tends to resemble sections of that text. This is also true of ordinary Markov devices whose probability matrices are calculated on the basis of a small text sample. As with such devices, it is to be expected that a larger text sample would lessen the resemblence of the output to any particular text without lessening its resemblence to the language in which the sample is written.

The first line of each pair is the input text, and immediately following it the corresponding output. For Markov filtering the input text is a randomly generated sequence of letters. The same input was applied to filters of different orders. (The input text is repeated each time to facilitate comparison.)

Markov filter:

```
mhcqlhogbvvcopxzyqdqmueo skjcsrl     (random input sample)
m cod oabapcopeloudemyet skict l     (order 3)

mhcqlhogbvvcopxzyqdqmueo skjcsrl     (random input sample)
m coth gbb comeloud mue  skjesse     (order 5)

mhcqlhogbvvcopxzyqdqmueo skjcsrl     (random input sample)
m coly ght jorthy locust skoes l     (order 7)

mhcqlhogbvvcopxzyqdqmueo skjcsrl     (random input sample)
m woly ghostop dord my mes the l     (order 9)

mhcqlhogbvvcopxzyqdqmueo skjcsrl     (random input sample)
e holy ghost many devils ahe spi     (order 11)

mhcqlhogbvvcopxzyqdqmueo skjcsrl     (random input sample)
m cometh one mightier at saw him     (order 13)

mhcqlhogbvvcopxzyqdqmueo skjcsrl     (random input sample)
e holy ghost and it even when te     (order 15)
```

The above example shows the effect of increasing order: each output is locally similar to English, but what is considered 'local' depends on the order of the filter. With the third order filter in the first example three letter sequences are similar to English, with unfamiliar consonant clusters like 'cql' being replaced with the more familiar 'cod'. Notice that although the output becomes more and more like English for the higher order filters, the process of increasing order is not convergent. The last three examples are not increasingly close to the same English text, rather they approximate different

texts. This probably reflects the size of the text sample stored in memory. A filter of order 15 compares strings of 15 characters, and the number of near matches in the stored sample may be very few.

The same program can be applied to text that is already in English. The examples below show what happens to an ordinary English sentence that is not a perfect match for any sentence in the stored text sample. Of particular interest is the word 'this' which begins the sample. With filters of order 3 and 5, the word is passed through unchanged. In fact the whole sequence 'this is a ' passed through the order 5 filter unchanged. Thus the filter of order five is a pseudo-identity filter: in the above example it altered a random sequence of letters to more nearly approximate English; here it passes a sequence from English unchanged. This is particularly interesting since the exact sequence 'this is a ' does not occur in the stored text sample. Because it does not, it does not survive the move to higher orders. With a larger text sample we would not expect such a familiar sequence of letters to be altered.

```
this is a sample sentence      (test sample input)
this is a samele sentence      (order 3)

this is a sample sentence      (test sample input)
this is a semplr sentance      (order 5)

this is a sample sentence      (test sample input)
thit is a shipleasentance      (order 7)
```

```
this is a sample sentence      (test sample input)
thit is a shiped sontance      (order 9)

this is a sample sentence      (test sample input)
thishired servants galile

this is a sample sentence      (test sample input)
thishired servants ahence      (order 13)

this is a sample sentence      (test sample input)
this ired servants and we      (order 15)
```

The above example showed a perfectly good English word 'sample' being changed into another, 'servants'. This happened because the word 'sample' was not sufficiently like anything in the programs memory. If a much larger text sample was provided, we could expect words such as 'sample' to remain unchanged.

This program can also be used as a corrective filter. If the input is very similar to a part of the text in memory, but does not exactly match anything, it will probably be changed until it is locally identical with the stored text. This can result in correction of typographical errors, as the following example shows:

```
and they weke both axtonished at      (text with errors)
and they weke both autonished at      (order 3)

and they weke both axtonished at      (text with errors)
and they were both astonished at      (order 5)
```

In this example the input was a string that matched a part of the stored text sample except for two errors. An order 3 filter did not correct these errors, since it only compares strings of three letters. For the input word 'weke', with an error in the third letter, the three letter strings 'wek', 'eke' and 'ke' are sufficiently similar to three letter strings in the stored sample. An order five filter compares longer strings, and therefore performs more correction.

An interesting further example relates to the question of learning, and has many implications, to be discussed later. This example starts with no text in memory, and adds each input line after having attempted to match it to whatever is in memory. Because the memory is empty to start with, the first input sample is matched by a string of blanks. The first and second sample have one word in common 'of', and this leads to a short match for the second sample. After that the sample more often finds matching strings, but the result is sometimes more like one of the preceding samples. Finally, after ten or more samples have been processed there is enough text in memory for reasonable matches to occur.

For all of the following examples a filter of order 7 was used:

```
the beginning of                    (start of input text)
                                    (blank line is only match)

the gospel of jesus christ the s
        g of       beginninning
```

on of god as it is written in th
g of the beginninnininninnin th

e prophet isaiah behold i send m
e gospelt ist th beginn jespel o

y messenger before thy face whic
jespel of beholf the beginning

h shall prepare thy way before t
the e prophre thy fah before t

hee the voice of one crying in t
hen the gospe of the peging of t

he wilderness prepare ye the way
hy wengermessengepare if thy way

of the lord make his paths stra
of the send mace ger besus chri

ight john did baptize in the wil
iahe gospegir bef ihe in the wil

derness and preach the baptism o
derness all prepye the baptize o

1.3.8 Feedback and Stable States

A serious problem with the pseudoidentity filter and the other linear devices described above is the one-to-one mapping of input symbols to output symbols. These devices perform as expected as long as the input symbols keep coming in at a predictable rate, but they are quite unlike the typical acceptor of automata theory, which just sits and waits for an acceptable input. The advantages of a device with stable states may be seen by examining more examples of simple filtering, this time as applied to continuous text.[12]

The text samples below illustrate artificial assimilation and dissimilation applied to Latin and Spanish, respectively. For Latin the filter coefficients are positive, indicating the same kind of low pass filtering as in the previous examples. The Spanish examples involve the use of negative filter coefficients; these produce high-pass filters, or synthetic dissimilation. For all of these examples the same filter coefficient is used for both dimensions.

---

[12] In these examples, as in the previous ones, I have used the hypen (-) to indicate a letter that has been transformed into a blank or word space character.

Original Latin text:

Facta autem hac voce convenit
multitudo et mente confusa est
quoniam audiebat unusquisque lengua
sua illos loquentes.

2-D model, filter coefficients: .075 .075

Facta autem cac poce convenit
multitoeo et mente confusa est
quoniam aoeiebat uousquisque linnoe
sua illos loquentes.

2-D model, filter coefficients: .1 .1

Faaoa aotem ccc -i-a conrenen
aulnetoeo et mente coofusa est
uuoniam aoeieman ooofuu-suu- -inno-
sua allos luuuentes.

As before we may notice certain very common words such as 'est', 'et', and 'sua' which are unchanged by the filter. Other words are more drastically changed. If we examine these changes, it should be obvious that the artificial assimilation presented here does not correspond to any possible assimilation such as

could have happened in the course of history. The filtered text has long sequences of vowels which are unlike those of any real language. The problem is not the actual assimilation, but the failure to compress sequences of vowels into shorter sequences or single vowels. The requirement that an output symbol must be produced for each input symbol prevents compression of sequences.

High-pass filtering, or artificial dissimilation, leads to the opposite problem, as the following example illustrates:

Original Spanish text:

Y hecho este estruendo se junto la multitud y estaban confusos porque cada uno les oia hablar su propia lengua.

2-D model, filter coefficients: -.075 -.075

Y hdcho este estrqendo se junto la multitqd y estaban confusos porque cada uno les oia hablar propia lengqa.

2-D model, filter coefficients: -.1 -.1

Y hdcho este estvqendo se jqnto la
multptqd y estaban cogfusos pfrqud
cada qgo yds oia hiblar su prfpia
yengqa.

Here we have consonant clusters that would be difficult to pronounce. Pronunciation of consonanant clusters is usually aided by insertion of transient vowel sounds, which often are lengthened into real vowels, as in Japanese borrowings of English terms. Excrescent vowels added to the filtered output given above would aid the pronunciation and produce something more closely resembling real speech. These examples clearly indicate a problem with the kind of filter previously described which produced exactly one output symbol for each input symbol.

The root of this problem in the lack of stable states. The acceptors and other automata normally used in modelling language have stable states, and the devices remain in each state until an appropriate input arrives. Inappropriate inputs will cause no change of state. In the next section I will discuss the the automata used in modelling grammars. These devices have stable states, and are more suitable models of actual linguistic processes. Unfortunately many of them seem to be non-linear devices. It will be argued that only linear automata are appropriate models of linguistic operations.

1.3.9 Rotation into the Time Domain

A device involving feedback and having stable states may also have unstable states. If so, it may respond quite differently to different inputs. Some inputs may simply cause a single change of state, producing a single output. Other inputs might cause no change of state and no output, while still others might lead to an unstable state, which will change spontaneously.

Let us consider a simple example using ordinary binary phonetic features. Suppose the input has the value 1 (or 'plus', as phonologists say) for each of the coordinates labelled 'strident', 'continuent', 'stop', and 'vocalic'. This combination cannot represent a sound in any human language, and so could be called 'ungrammatical'. A simple projection could be designed to replace it with a feature vector representing a possible sound, but which one? It could be 's', (because of the stridency), or 't', (a typical stop), or a continuent such as 'r', or the most common vowel, 'e'. But choosing any one of these would involve a loss of information. The alternative is to produce a sequence 'stre' which contains each of the various sounds in a predictable order.[13]

This latter situation can be important in modelling some aspects of language. A problem with grammatical filters was mentioned earlier in discussing the pseudo-identity filter. This

[13]This idea was brought to my attention by Dr. E. W. Roberts. See Roberts(1972).

filter is technically a projection, and thus seems to involve a loss of information. How can we design a filter to correct or reject ungrammatical inputs that will not lose information?

An alternative to projection is rotation in time. A feature vector representing a single symbol has no coordinates representing time. Implicitly, this symbol occurs at an instant in time. If we add to each set of coordinate values a single time coordinate, then instead of considering an utterance as a feature matrix, or ordered sequence of vectors, we may choose to consider it as an unordered set of vectors. Translation from the unordered set back to the ordered sequence just involves placing the vectors in order of their time coordinate.

A filter that is not a projection but a rotation involving the time coordinate could be applied to a set of vectors, and after it is applied, the unordered set could be translated into a sequence of vectors, and then into symbols.

If the filter is designed properly, its output will always be a grammatical sequence. To design a such a filter we can first produce a filter that projects its input onto a hyperplane containing only grammatical sequences. The method described previously for creating a pseudo-identity filter can be employed for this purpose. Suppose we now take the output of the projection and compare it with a (similarly delayed) copy of the input that produced it. The result is a vector containing only those components suppressed by the projection. If we add these components to the current input, the information they contain will not be lost.

PART TWO

Problems in the Development and Application of Linear Models

In this part I discuss difficult aspects of the theory and subtler details of the experiments, including short sections on a number of topics related to the earlier material.

First is a discussion of the relationship between the present linear filtrative theory and generative grammar. The difference between filters and devices from automata theory is reviewed and it is shown that filters can model all of the others, and so are the most general approach.

Adoption of a filtrative model is possible without accepting the restriction to linear filters, but even if one imagines nonlinear filters, there are advantages to using the terminology of linear operator theory to analyse them. An algebraic basis for doing this is provided.

The next section of Part Two gives biological and physical models of the brain based on linear devices, attempting to show by example how human beings could be linear. In this section I attempt to meet objections that might be raised by scientists, engineers, or mathematicians familiar with linear devices. One objection is that each of us has some training in arithmetic which enables him to square numbers and perform other nonlinear operations. How can we do this if the human brain is linear? The answer is to be found in an account of simulation, showing how

linear devices can be constructed to simulate nonlinear operations with arbitrary accuracy, at the cost of great inefficiency.

The last section discusses additional details of the experimental work, including different kinds of text samples and different languages. This discussion is also intended to show how the methods can be extended and improved.

2.1 Generative and Filtrative Grammar

Almost every linguist has his own theory of language, but most of these seem to be variants of Chomsky's generative grammars, which are based on the notion that individual human beings follow rules to generate utterances (Chomsky 1961). Many of the theories of generative grammars include both generative and filtrative components, with filtrative processes acting on the output of the generative component (Chomsky 1965:139). The theory to be presented in this thesis leaves out the generative component altogether while emphasizing the role of filtering, and so could be called filtrative grammar.

The notion of a grammar as a filter serves to correct a serious conceptual error that I believe has limited the development of linguistics in recent years. Certain older theories such as Immediate Constituent analysis have been prematurely abandoned on the basis of mathematical results that I find irrelevent and misleading. Their adoption as standards of proof in linguistics has led away from a view of language that was compatible with known psychological and biological theories towards a treatment of language as an abstract calculus or logic with improbable properties, that could not be learned, but must be largely innate.

Evidence that generative grammar may still be an unscientific discipline may be found in the behavior of

contemporary linguists, many of whom spend a lot of time and effort in relatively informal argumentation, often aimed at convincing their opponents of the relative superiority of their own particular version of generative grammar. In recent years many linguists have described these disagreements as being over differences in notation, with some suggesting that two or more proposed theories are just notational variants of one another, and others denying this.

The question of whether or not two notational devices are equivalent is a mathematical question, and need not interest the linguist at all. If this discipline had reached the mathematical stage in the development of its formal notation, one would expect that mathematical proofs would be the common method of settling purely notational questions. It seems inevitable that the current state of affairs will be replaced with one which is more satisfactory, (although perhaps less enjoyable), characterized by the rigorous mathematical treatment of the kinds of problems that give rise to disputation today.

How might this change to a more scientific discipline come about? In the next sections I examine various approaches to formal linguistics that have already been developed. There are two related ways of linguistic analysis:

1. The analysis of the speaker of language, or the listener, or both, as formal devices with certain capabilities.
2. The abstract treatment of language as a mathematical structure, without reference to the speaker or listener.

This approach involves treating language as a kind of logic or algebra.

In what follows I will describe both approaches, and the parallelism between them.

2.1.1 Devices Used in Modelling Language

To describe the first approach I will make use of information theory and the notion of information flow. Proponents of generative grammar are vague about the flow of information at the generative level. The branch of generative grammar known as Generative Semantics is most specific. In this version of generative grammar, also known as 'semantic syntax', the input to the collection of rules and transformations is some semantic representation. For some generative semanticists, the semantic representation is taken to be similar or identical to the first-order Predicate Calculus of formal logic. If so there must be some process by which a person's meaning is encoded in this logical form prior to the linguistic processes that produce an expression of it in a natural language, but this process is not discussed.

In other versions of generative grammar, such as the 'Standard Theory' proclaimed by Chomsky, or the 'Extended Standard Theory' and other extensions and simplifications that followed it, there is some level known as the 'base', to which transformations apply. The base is generated by phrase structure rules, but does not represent meaning. Different bases are

generated for different purposes, and are transformed in different ways, by the selective use of optional rules and transformations. Presumably the input or source which directs the choice of rules and transformations is some component of the human mind, but its exact role is unclear.

In making mathematical models and proving theorems about generative grammars, many linguists seem to assume the existence of some kind of stochastic process that controls the application of optional rules or transformations. Chomsky refers to an early theory of grammar as the "theory of finite-state Markov sources", and regards it as disproved. This theory is based on a finite-state machine driven by a stochastic process. It could be argued that the input to a generator is stochastic, not in the sense of being random, but in being uncorrelated, since the speaker ought to be able to say anything, or generate an expression of any meaning whatsoever. I suspect such a notion is at the root of generative grammar, but this is not clear from the literature. Indeed, it is not clear why generative grammar has such a name, since the actual generative processes are so poorly described.

I suggest that all of these theories are in fact filtrative theories. In writing phrase structure and transformational grammars, linguists only concern themselves with writing rules, and not with the element of choice or chance that determines how these rules will apply. A system of rules is only made into a generative system when some system for choosing and applying the

rules is specified. Linguists do not bother specifying how rules are chosen, and so have not actually developed a generative grammar.

Nor would there be any basis for doing so. I can see no justification for assuming any particular kind of generative source or input. The most we can do is describe a formal device that processes an input, and this is best described as a filter.

There are actually two filters at least that need to be described, unless one is going to adopt a behaviourist approach, in which case we would describe one combined filter. If we follow Chomsky's "mentalistic" approach, then we must describe an input filter that processes speech and produces or induces mental states, and a output filter that expresses mental states in speech.

In fact, generative grammarians do describe two kinds of grammatical devices, generators and acceptors. Acceptors are clearly some kind of filter, although the term is not usually applied to them, since they accept an input and produce a corresponding output. It would be easier to describe these acceptors if it were explicitly stated what output they produce is for a nongrammatical input. Ordinarily the output of a finite-state acceptor is taken to be its current state, and the devices do not change state except on receipt of a grammatical input. But in actual implementation of acceptors, (in computer software), ungrammatical inputs send the automaton to a special error state.

As I have indicated in an earlier section, generators are also implemented as filters, applied to a random noise source. The difficult question is whether or not these various filters are linear or nonlinear. Formal languages defined by phrase structure rules may be ambiguous in that the same string of terminal symbols may have two or more different derivations. It follows that the formal devices corresponding to these grammars may be nonlinear. I suggest that only linear automata are worthy of being considered as linguistic devices. It remains to be shown that linear automata can be developed to play the normal roles of automata in modelling grammatical devices. How can these properties be simulated with linear devices?

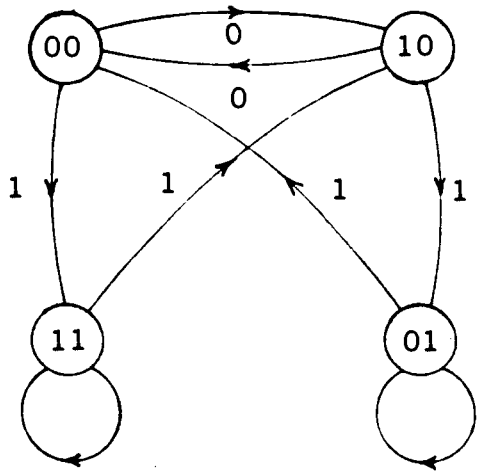
The fact that a linear device can have several distinct stable states may come as a surprise to anyone familiar with the more common linear devices used in electronics. Such devices often have just one input, and unless they have memory or delay elements in them, have only one stable state. In fact the number of stable states of a device depends on its order, and on the number of feedback loops. At most, the number of stable states is the same as the order of the device. For there to be any stable states at all, there must be feedback. For devices that accept vector inputs of order greater than one, it is not so easy to label feedback as negative or positive, but it is possible to say that when the device is in a stable state, the feedback is negative or homeostatic, i.e. tending to bring the system back to that state. Rather than using the delay, we may

choose another form of output that will let us produce a multistable device. Let the output be a vector made up of two sets of components, one of which is itself a vector representation of the last acceptable symbol, and one of which is a representation of the current state of the device. Let the components representing the current state of the device be fed back to the input. The operation performed on the combined input vector can be chosen so that the resulting device with feedback has several stable states, and will move between them only on receiving appropriate inputs. Figure 12[1] shows a simple finite state machine and how it may be implemented with a linear device.

Let us try to describe an automaton that will function as an acceptor for strings of alphabetical symbols. There are several problems of definition to be solved. Such an automaton will change state if it receives an appropriate symbol. Otherwise it will stubbornly remain in whatever state it was in before receipt of the symbol. Thus if the symbol "B" puts the device into state 47 and the device is one designed to accept English phonology, then the receipt of the symbol "Q" while it is in state 47 will cause no change of state, since "BQ" is not normally found in English. In fact "BQ" may appear in printed text, as in the term "subquote". Some people would be more inclined to write this term as "sub-quote", with the "-" indicating a morpheme boundary. This is a vital point, and the

---

[1]adapted from Albus(1975)



Finite-State Machine

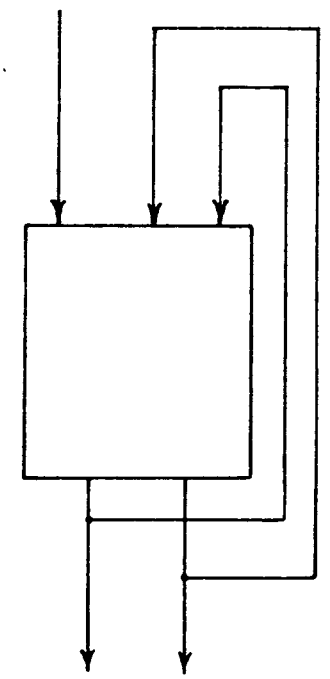
Transitions between states are indicated by arrows

Inputs values are written beside the arrows, and the outputs, or state vectors, are indicated as numbers inside circles.

Input Data	Current State	Output State
p	q r	s t
0	0 0	1 0
0	0 1	0 0
0	1 0	0 1
0	1 1	1 1
1	0 0	1 1
1	0 1	0 1
1	1 0	0 0
1	1 1	1 0

Another way of describing the same device is to give a list of all possible inputs, together with the corresponding outputs for each state.

Can the input/output list shown here be implemented as a linear device?



Yes. All that is necessary is a vector representing the current state. This can be obtained from the previous output as feedback previous output.

Let: input data = p
current state = q,r
output state = s,t

The following system of linear equations over the binary finite field specifies such a device:

$$s = -(q + r)$$
$$s = p + q$$

Here '+' represents the Exclusive-Or operation which is addition in the binary finite field.

Figure 12.Linear Implementation of Finite-State Machine

key to more general application of linear automata: boundary symbols can be generated to permit the machine to output a signal and change state. I believe natural language contains boundary symbols such as pauses and changes of tempo, and that these are properly written in text as punctuation symbols, which are vital to the understanding of written material. This is discussed again in the next section.

If we do not have an intervening punctuation symbol, "BQ" will not be accepted, and so the receipt of "Q" after "B" will not cause a change of state. On the other hand, the symbol "S" may follow "B", so its receipt will cause a change of state.

2.2 Language as Algebra

The primary purpose of this section is to show the deficiencies of current algebraic treatments of language, and to suggest an adequate algebraic formalism. Such a formalism can be adopted whether or not one accepts the filtrative model of grammar, and without any committment to linear models.

It has often been noted that generative grammar is based on a kind of algebra, but the current state of linguistics is quite unlike the orderly discipline that is modern algebra. It is more like the difficult discipline of formal logic, which is splintered by philosophical disputes. This is not coincidental, since the course of modern linguistics has been very much influenced by the work of logicians.

Chomsky's original theory of transformational generative grammar was developed in Logical Structure of Linguistic Theory (Chomsky 1955). This early work does contain some applied logic, but it makes no use of the algebraic theory of formal systems and languages. Chomsky seems to have based his new theory of grammar on formal devices such as used by logicians, without being aware that many logicians were reexamining their work on an algebraic basis. It was only later that Chomsky attempted to analyse the algebraic implications of his work, and by that time he was committed to a form of algebra that presents enormous technical problems.

In order to illustrate what a difficult branch of algebra has been employed by generative grammarians, it should suffice to note some of the theorems proved about phrase-structure grammars and transformational grammars based on them (see Chomsky 1963:382ff). These so-called "undecidability theorems" demonstrate that the solution to certain problems in algebraic syntax depends on the skill of the mathematician and may require years of research, which may not produce any results. There are theorems showing that it is undecidable whether or not a context-free grammar generates all strings of its terminal elements. Also, it is undecidable whether or not two such grammars generate the same language, or whether the language generated by one grammar is a part of that generated by another. Thus there is no algorithm for deciding whether or not a context-free grammar is trivial, or whether two such grammars are just notational variants of one another. It is still a mathematical question whether or not two such grammars are equivalent, but there is no systematic way of attacking the question with guaranteed results.

Chomsky's early work, and indeed the work of linguists preceeding him, made use of algebra. But Chomsky apparently made no attempt to employ the vast theoretical repertoire of the algebraist in selecting an appropriate formalism. Instead he invented a formalism, and only later made an attempt to apply algebra to it. The thorough application of existing theory might have led to a more fruitful approach.

I believe that a thorough and systematic attempt to apply algebra to the study of language would lead to a treatment of grammar within the theory of linear operators. The current use of algebra in linguistics involves much weaker systems with few of the formal properties that make linear operator theory so easy to use. To illustrate the differences between algebraic theories it may be useful to discuss a few of the more basic properties in some detail.

Let us begin by considering an algebraic structure formed from a human language in the most natural way. We may note that the vocabulary of most human languages is a finite list of words, and that sentences are strings of these words. We may also note that there are constituents of sentences which consist of shorter strings of words, called phrases. All of these phrases in a language can be made by juxtaposing words in the language, and all of the sentences can be made by juxtaposing words and phrases.[2]

The most natural algebraic structure is probably the one which includes as elements all of the words, phrases and sentences of a language, and has juxtaposition as an operator. The most important properties to consider are closure, associativity, and commutativity. We might also attempt to define a second operator related to juxtaposition as multiplication is related to addition by the distributive

---

[2]We might also consider the formation of words from morphemes, but this involves a few problems and is better left for a later stage.

property. Closure and associativity are so important to the solution of algebraic problems that I would have expected that any algebraic approach to syntax would abandon them only after considerable study had proved they could not be retained. However linguists seem to have attached little importance to closure and associativity, as if they had some deep seated belief that they were irrelevent to human languages. On the surface, this might seem to be an entirely reasonable belief, and it is easy to construct an argument supporting it.

The closure property demands that an operator be applicable to any two elements of the algebraic structure, and that the result of the operation be another element of the structure. But juxtaposing words and phrases at random will not always produce phrases and sentences in the language, so it seems that the closure property does not apply to such structures.

Next consider associativity. Suppose that we consider each word or phrase to have a particular meaning. The phrase 'cars and women' refers to a union of two large classes, and we wish it to retain that reference even when combined with the word 'fast' and the phrase 'are interesting' to form a sentence listing a playboy's passions and pastimes. But the sentence 'Fast cars and women are interesting' could either be the sentence formed by combining the elements mentioned above, or the sentence formed by combining the phrase 'Fast cars' with 'and women', and 'are interesting'. We know that these are two different sentences, one about fast cars and fast women, and one

that includes less cooperative women as well. But to say that these are two different sentences is to deny the associative property, according to the way in which the strings are grouped before juxtaposition has no influence on the string formed by juxtaposing them.

It seems on first inspection, therefore, that the two properties of closure and associativity must not apply to human languages. Indeed, the common notion of syntax seems to depend entirely on the absence of these properties.

It is due to this supposed lack of closure that linguists can claim that the object of syntax is to generate the sentences of the language and only the sentences of the language. If closure applied then all combinations of words would be allowable sentences, and grammars would differ only in their vocabulary. Similarly, the lack of associativity seems essential to syntax. We know that a phrase structure grammar is formally equivalent to immediate constituent analysis, in that both types of syntax give an analysis of the sentence into smaller and smaller constituents. The constituents of a sentence are the phrases from which it is composed, and the constituents of a phrase are smaller phrases or words. Sentences which contain the same words in the same order are considered to be different if the groupings of words into phrases, and phrases into sentences, are different. Yet this is just the denial of the associative law.

The associative property may be extended to the cases that formerly told against it with the aid of Chomsky's distinction between competance and performance. It should be noticed that pairs of sentences such as 'Fast (cars and women) are interesting' and '(Fast cars) (and women) are interesting' are often pronounced differently. We can disambiguate these sentences either by inserting a slight pause to mark the parentheses shown in the above examples, or by speaking the part of the sentence within one set of parentheses at a slightly accelerated tempo, or by using both of these methods at once. The fact that we have the ability to disambiguate sentences by careful attention to prosody indicates that ambiguity is a matter of linguistic performance rather than competance. Poor perfomance can leave the sentences ambiguous, but it seems that linguistic competence includes a means for disambiguating sentences by giving verbal clues to their internal structure.

Most mathematicians would probably agree that if there is any practical way of redefining a problem to make the associative law apply, this should be done. The simplest way to accomplish this is to add a slight pause to the basic vocabulary of the language. We could thus rewrite the two different sentences as two different strings of words:

Fast (pause) cars and women are interesting.

and

Fast cars (pause) and women are interesting.

This solution amounts to the assertion that human languages incorporate punctuation markings in an essential way, and is thus not a comma-free code. As noted before, the addition of punctuation symbols greatly extends the abilities of finite state machines, and makes linear models of grammar an easier task. The fact that these punctuation symbols are prosodic, rather than distinct symbols does not change this. In fact, words such as 'that' and 'which', or phrases employing them, may be seen as little more than punctuation symbols, serving the same role played by parentheses in computer languages.

2.2.1 Closure

Closure presents an entirely different kind of problem, but fortunately it is one that has a familiar treatment within mathematics.

Mathematicians make extensive use of an artificial number system called the complex numbers, which has the property of closure, even though data from the real world is always in the form of real numbers. The system of real numbers is not algebraically closed, so a related closed system is used instead. This involves inventing a whole new class of numbers called imaginary numbers, which are used when the results of operations on real numbers have no real numbers as results. For example, the operation of square root is often used in the real number system. Yet no real number is the result of taking the square root of a negative real number. Imaginary numbers fill

the gap by providing the square roots of negative numbers, so that the system of real and imaginary numbers becomes closed under the operation of square root.

Although data from the real word is in the form of real numbers, physicists and others who use the mathematics of complex numbers have found that this number system is not only more tractable mathematically, but also provides insightful formulations of phenomena in their discipline. This is a familiar effect: a sophisticated new notation with "nice" mathematical properties is often found to be a source of insights about the subject matter to which it is applied.

It would seem reasonable to investigate the possibility of obtaining closure in a syntactic algebra by allowing imaginary sentences. Thus, we could consider all combinations and permutations of the vocabulary as sentences, hoping that some natural criteria will emerge to distinguish the real and imaginary sentences. This approach could be more fruitful than the use of phrase structure rules or other such devices that lead to unwieldly mathematical formulations.

If the formalism employed to study language allows the consideration of all possible strings of symbols, acceptable and nonacceptable, then algebraic closure is achieved.

If closure and associativity hold, then by definition, each word in a language is an element in a semigroup, with the operation of juxtaposition forming other elements of the semigroup. Rather than dealing directly with semigroups, it is

possible to find another kind of structure in which any semigroups serving to model natural languages can be embedded.

When one system or structure is embedded in another, the larger system may be a mathematically interesting one with very desirable properties, while the embedded structure may have only those properties that are needed to reflect or model some aspect of reality. The problem of embedding an algebraic structure to be studied within a more convenient kind of structure is known as the representation problem.

If one structure is embedded within another, the term 'extension' is often used, as when we say that the complex number system is an extension of the real number system. In order for extension or embedding to be possible the two kinds of structure must share certain properties. The term 'hereditary property' is sometimes applied to properties which must apply to an embedded structure if they apply to the structure embedding it. Closure is not a hereditary property, but associativity and commutativity are.

If language is indeed associative then we may choose to model it within an associative algebraic structure, such as the algebra of matrices over the real or complex numbers, or the algebra of linear operators. [3]

These algebras have two operators, addition and multiplication. Addition is commutative in the algebras of matrices and the algebra of linear operators, but multiplication

---

[3] The algebra of linear operators includes the algebra of matrices as a proper subalgebra.

is not. We may embed the noncommutative semigroups discussed above within the multiplicative group of either of these algebras, since the existence of inverses is not a hereditary property.

To create such an algebraic model of language, each word is taken as a symbolic representation of some element of an algebraic structure. It is important to realize that in any mathematical model of human language we will need to employ very large structures. Using single words we can refer to only some of these elements. To refer to the others we need to concatenate words into phrases and sentences.

In suggesting that linguistic items be formalized as linear operators, I am not committing us to any particular formalization or model of the human language user who manipulates these operators. Nevertheless, there are interesting reasons for adopting a compatible model of the language user. It is common in formal language and automata theory to consider that there are types of languages and corresponding types of automata that accept or generate them. We often use the same term to describe the language and the automaton.

There is a well known correspondence between languages and automata:

- Unrestricted phrase structure grammars correspond to members of the class of Type 0 languages (or recursively enumerable functions), and the class of Turing machines.
- Context Sensitive grammars correspond to members of the

class of Type 1, or CS-languages, and the class of linear-bounded automata.[4]

- Context Free grammars correspond to members of the class of Type 2 or CF-languages, and the class of pushdown automata.
- Regualar or Finite state grammars correspond to members of the class of Type 3 or regular or finite state languages, and to the class of finite-state automata.

Chomsky's transformational grammars used transformation rules applied to structures generated by Context-Free phrase structure rules. The class of languages definable by transformational grammars of this type has been shown to be the class of Type 0 languages. This result has caused much disturbance in the TGG community, since it was originally suggested that the class of possible human languages lies somewhere between Types 1 and 2 (CS and CF languages.)

The arguments presented in this thesis are intended to show that this nice set of correspondences is quite misleading, and entirely irrelevent to the study of human language. All of the classes of machines listed above can be treated as filters, by treating their variables as inputs. If this is done, it may been seen that among all classes there are both linear and nonlinear machines. My view is that only linear machines correspond to languages.

What I have called variables, are treated by linguists as category symbols, and are important to them as giving the

[4] Linear in this term means one-dimensional, and does not correspond to the term linear as used elsewhere in this thesis.

structural description of the generated sentence. The structural description of a sentence includes many things, but its most essential aspect is an account of the grouping or bracketting of components. I have argued that this is an error, and that brackets should be generated as terminal symbols.

In this regard it is worth considering the use of brackets as symbols in set theory. This use is consistent with their use as grouping devices in linguistics. Yet pure abstract set theory is often formulated so that its only terminal symbols are parentheses. An empty set of parentheses indicates the null set, and all other mathematical objects are ultimately may up from combining sets containing the null set into elaborate structures. In set theory a string of parentheses such as

((())((())(())))

is a well-formed expression standing for some mathematical object.[5]

Chomsky's theory of grammar involves more than mere bracketting: he uses tree structures, or their equivalents, labelled brackets. I believe that this is not necessary. Some indication of bracketting is important, but the attribution of grammatical labels may be treated differently. According to Firth, grammatical categorization is a form of meaning, since the concepts of 'noun' and 'verb' are abstactions of common

[5]This is easier to see if blanks are added:
((()) (((())) (())))

meanings. It would follow that the labelling of brackets is a part of the process of decoding language, the inverse of the encoding process described by Generative Semantics. I suggest that there is some linear filter which can decode the meaning of utterances, and a simplified version of this filter could be used to provide grammatical labels. This aspect of parsing should thus await the development of filters for encoding and decoding. I believe such filters are easily developed as linear implementations of Markov filters, and their inverses.

The problem of creating filters to perform a specific task is discussed in a later chapter, as part of an analysis of how the human brain may solve the same problem.

2.3 Creating Linear Devices for Specific Purposes

Consider how such an acceptor might be simulated with a linear device. The first problem is that we have no actual output to express as output vectors. There are several possibilities:

1. The output could be zero or one, indicating acceptance or rejection of the input.
2. The output could be a symbol representing the current state.
3. The output could be the last acceptable input.
4. The output could be identical to the input if it was acceptable, and zero otherwise.
5. The output could be the nearest acceptable sequence to the input.

In extending the theory of acceptors to linear devices, we may choose whichever of these alternatives seems most appropriate. Or we may have several outputs. One factor that may help determine our choice of output is the possible need to feed some or all of the output back to the input. By considering this possibility, we can make a preliminary choice from among the last three of these cases, since in any of these cases the output is an alphabetical symbol like the input.

Suppose the input is a single alphabetical symbol, and so is the output, which is normally identical with the input. If we merely perform the process described above, we will produce

something very much like the identity matrix. Obviously the identity matrix cannot serve as a linear acceptor. One solution is to make the input and output a sequence of two or three symbols. Some of the possible sequences of two or three alphabetical symbols are acceptable in English, and others are not. Using a well known algorithm, to be described in a later chapter, a matrix can be produced which will transform a vector representation of the input sequences into whatever form of output sequence has been chosen.

But will this produce a genuine linear acceptor? The acceptors described in automata theory take in single symbols, not sequences of symbols. Can we not produce a linear device that will take in just one symbol at a time and either accept or reject it? In order to do so, we may employ a device that introduces a time delay. As shown in the diagram of the Pseudo-Identity filter in the next section, we may accept a single symbol, copy it into three identical inputs, and then delay one of them. The resulting three inputs are then a sequence of symbols.

2.3.1 Autocorrelation Filtering and Pseudo-Identity Filters

The creation of a linear grammatical device involves the creation of a filter which will pass grammatically correct or acceptable text without changing it in any way, but will perform some correction on any other input. Ideally the output of the device should be grammatically correct no matter what the input

is, with small deviations from correctness being changed in a natural way. Such a device would have practical applications in word processing where it could be used to check and correct spelling and/or grammar without the need of large dictionaries and lists of rules.

The requirement that the output of the device be grammatically correct for any input cannot be easily met, but it can be approximated for a wide range of inputs. There are two types of input that must be considered, input that is already grammatically correct, and random input. The former will be correlated input, the latter uncorrelated. Thus the device must impose a correlation on the input if it does not exist already, but must not significantly alter the input if it is suitably correlated. Let us consider that the correlation of grammatically correct material is represented by an autocorrelation matrix. An autocorrelation matrix can be used as a filter, and will impose its own correlation on random input, thus satisfying one of the requirements. But what will it do to input that is already correlated? The autocorrelative filters that produce approximations or simulations of English from a random input are not what is wanted for corrective filters, since they produce very strange looking text when given ordinary English text as input. The reason for this can be seen in analysing these filters.

If input with a certain correlation is passed through an autocorrelation matrix representing the same correlation, the

correlation of the resulting output would be as represented by the square of the autocorrelation matrix. This may or may not mean that the correlated input is altered. A matrix may be equal to its square if it is a matrix representing a projection. This leads to the conclusion that a grammar is a projection of the space of possible linguistic forms onto the subspace of acceptable forms.

To see what an autocorrelation matrix does do to its input vectors, I used a program to calculate the eigenvalues and eigenvectors of an autocorrelation matrix.

The eigenvalues for autocorrelation matrices that acted as linear Markov filters tended to cluster near zero and one, with a few eigenvalues in between. Thus the effect on the eigenvectors tended to be one or the other of two actions, to leave the eigenvector almost unchanged in magnitude, or to almost annihilate it. This suggested the creation of a matrix that would be based on the autocorrelation matrix, but would do just one or the other of these operations exactly.

To form such a matrix the following algorithm can be used:

1. Create an autocorrelative filter and calculate its eigenvectors and eigenvalues.
2. List the eigenvectors corresponding to eigenvalues above some cutoff point.
3. For each of these eigenvectors create a matrix by multiplying the vector by itself. Such a matrix has rows that are identical except in magnitude, being scaled by

values of successive coordinates in the vector, and its columns are just like its rows.

4. Take the direct sum of these matrices by adding them point by point.
5. Normalize the matrix.

The resulting matrix describes a projection onto a hyperplane containing only the chosen eigenvectors. Vectors consisting of linear combinations of those eigenvectors (only) will be unchanged by the transformation. The fact that such a matrix does pass English text through unchanged amounts to a definition of English grammatology, since it implies that all correct sequences of letters in English can be represented as linear combinations of a few letters, and are thus contained on a hyperplane embedded in the space of all possible sequences of letters.

2.3.2 Towards Linear Corrective Filters

By using the algorithm just given, a particular matrix can be described which will serve as a linear corrector and will produce only grammatically acceptable or syntactically correct strings of symbols. This is similar to a type of filter used to remove noise from data. The pseudoidentity filter just described is a member of a class of filters which approximate more and more to the ideal of a linear corrective filter.

It is worth considering linear corrective filters in more detail. The usual application of a linear corrective filter, or

linear corrector, can be described as follows: Given a linear corrective filter of order n , and n samples of noisy data representing successive values of some coordinate, the output of the filter is an estimation of the correct value of one of the n values, usually the first or last one. This estimated value may be used with other corrected values in an attempt to further improve the data. If the filter is used recursively, the quality of each successive correction is lower than its predecessor, and there is some danger of the device being unstable.

An interesting property of such corrective filters is that they have no effect at all on some functions. Analytic functions are a class of functions that are exactly correctable, and a linear corrective filter that matches a given analytic function will pass it unchanged. Since the filter is based on the autocorrelation function derived from a typical data sample, and since a large number of different functions may have the same autocorrelation function, it seems that these filters will pass many different functions without having any effect on them.

Although the usual application of a linear corrective filter is in correcting the next value of a changing variable quantity, and may be used with a feedback loop so that this corrected value is used in correcting subsequent values, it is possible to use the filter in a much simpler application in which it will convert a long input function into a long output function that resembles it more or less closely. Each value of the output function (for a filter of order n) will be a weighted

sum of n values of the input function. The weights are so chosen that for certain analytic functions the weighted sum of n successive input values is just the next input value. If random noise is added to one of these analytic functions, the sum of n successive input values will be equal to what the next value of the analytic function alone would be, plus the weighted sum of the random noise values. If the random noise had zero mean, then this part of the weighted sum would probably be very near zero. Thus the filter would tend to remove the noise while leaving the analytic function unchanged.

Unfortunately, the pseudo-identity filter produced by the above process is not a good linear corrective filter. Although it passes English text through unchanged and alters other sequences of letters, it does not always change them into correct English sequences. How close it comes to this goal depends on the size of the matrix, in a way that is independent of the number of letters represented in the vectors it handles. The linear corrective filters being prepared by the algorithm just given can also be partitioned into blocks in which each block is a cross-correlation matrix for pairs of letters. The accuracy of the filter in correcting non-English sequences into English or pseudo-English depends on the size of these blocks more than on the number of blocks in the matrix.

Analysing autocorrelative filters with varying block sizes showed that the eigenvalues of such matrices are more closely clustered around the two extreme values when the matrices have a

large block size. This may be explained in the following way: The eigenvectors of the smaller matrices are related to the eigenvectors of the larger matrices in a simple way, since both sizes of matrix are based on similar sets of vectors. There is actually a homomorphism between the large vector space on which the larger matrix operates and the smaller space of the smaller matrix.

For example, consider two autocorrelation matrices which are based on vectors of size 8 and 16, respectively. Suppose they both act on vectors representing pairs of letters. Each single letter block of the larger matrix is an eight by eight submatrix, and each block of the smaller matrix is a four by four submatrix. There are many ways of defining a homomorphism from the space whose vectors have 16 components into the smaller space with 8 components to each vector. The simplest of these simply maps pairs of coordinates in the larger space onto single coordinates in the smaller by taking the arithmetic mean of each pair of coordinates. For example, the two rows of numbers given below might represent vectors in each of the two spaces:

0 0 0 1 1 0 1 1 1 1 0 1 1 0 0 0

0 .5 .5 1 1 .5 .5 0

Each value in the second vector is the arithmetic mean of two

values from the first vector, those above and slightly to either side of it. The first vector has only binary values, while the second has an intermediate value: the arithmetic mean of 0 and 1.

The same homomorphism that transforms vectors in this way must be applied to matrices, and is similar in effect. The blocks of the larger matrix will be mapped onto blocks of the smaller, with two by two subblocks being mapped onto single values of the smaller.

Autocorrelation matrices defined on the larger space will have 16 eigenvectors, while the those of the smaller space will have only eight. This is because two distinct eigenvectors of larger matrix will be mapped onto single eigenvectors of the smaller.

This mapping of pairs of eigenvectors onto single ones affects the clustering of eigenvalues. If one eigenvalue of the larger matrix is nearly one, and another is nearly zero, and this pair is mapped onto a single eigenvalue of the smaller matrix, that eigenvalue will have to be intermediate between the extremes, as in the above example. Thus if we find a matrix whose eigenvalues are spread over a range of values, we can postulate the existence of an equivalent matrix (defined on a larger space) whose eigenvalues are clustered around two extremes.

I suggest that the difference between autocorrelative filters and linear corrective filters is the distribution of

their eigenvalues. If the intermediate eigenvalues of autocorrelation matrices can be treated as resulting from the combination of eigenvalues from a higher space, then the use of larger spaces would lead to autocorrelation matrices that are more nearly usable as corrective filters.

Ideally we should consider a space of very large size in which the autocorrelative filter and linear corrective filter are the same since there are no intermediate eigenvalues.

There is a simple and natural way to ensure that the underlying space is of this form and that there are indeed no intermediate eigenvectors, and that is to base the linear model on the binary finite field.

2.3.3 Binary Linear Models

It must be noted that the property of linearity is definable for the transformations and operators of vector spaces over any field. The usual fields for any work with vector spaces are the field of real numbers and the field of complex numbers. Experience that is limited to vector spaces over these fields may mislead one to believe that continuity is an important part of linearity. Thus, for example, one may believe that any linear operator must have some representation as a continuous straight line. This is not at all true. A linear operator may be defined on a vector space over a finite field, and all of the values of any components of the vectors in that space will be one of the finite set of allowable values. An example of this that is very

important in what follows is the use of the smallest finite field in defining vector spaces.

The set containing only the two numbers zero and one is the basis for a field that can be defined in either of two ways. It has been shown that such a field must use two binary operations whose Cayley tables are essentially the truth tables for the 'exclusive-or' and the logical 'and'. However, the number zero does not have to be the additive identity it usually represents, nor does the number one have to represent the multiplicative identity. These roles can be reversed, in which case the two operations that look like the logical 'exclusive-or' and the logical 'and' are actually the logical identity and the logical 'or'.

These represent the only two different interpretations for the formal symbols of the smallest finite field, they are equivalent. For most of what follows we shall assume the more usual interpretation, in which the operation of addition is represented by the 'exclusive-or', and the operation of multiplication is represented by the logical 'and'.

This field bears an interesting relationship to formal logic. It has become customary in computing science to use the term Boolean for any system that uses only the numbers zero and one. This usage is contrary to history. Various algebraic structures based on zero and one were known before George Boole invented Boolean algebra. His contribution was not the restriction of the underlying set to two values, but the

definition of a new type of algebraic structure that used the two operators similar to logical 'or' and logical 'and'. Logical 'and' is often called logical multiplication since its role in a boolean algebra is analogous to the role of multiplication in the real numbers, as it is in the smallest finite field. Where boolean algebra differs from the field built on 0 and 1 is in the operator known as logical addition. Boolean algebra and propositional logic use the nonexclusive 'or' operation as a kind of logical addition, whereas the smallest finite field uses the 'exclusive-or'.

The investigation of the binary finite field may lead to genuine linear corrective filters, which do more than just recognize text, but will actually correct it. I have made some progress in this direction already and look on it as the next avenue of research.

The binary finite field and its extensions pose a difficult methodological problem. In constructing filters to serve a specific purpose we often want to <u>average</u> or perform a direct summation of simpler filters. In doing this we often make use of the notion that unwanted additive noise in a variety of simple linear devices will cancel each other out when the devices are averaged or added. This is not true for linear devices defined on fields of characteristic two, such as the binary finite field. The same problem is found in other algebraic structures. To solve it, we need to use some version of the optimization method described earlier.

One serious question remains, and that is the adequacy of linear models. The human brain is generally assumed to be a very powerful information processor. Are linear devices capable of performing the operations we know the brain can perform? Each of us has some training in arithmetic which enables him to square numbers and perform other nonlinear operations. How can we do this if the human brain is linear?

Although it is difficult to say for sure what a linear device cannot do, it is easy enough to list some operations familiar in information processing that are easily accomplished with nonlinear devices such as digital computers, but are far beyond the normal capabilities of linear devices:

1. Storing information by address to be recovered after an arbitrary interval of time.
2. Performing a processing loop a certain number of times depending on a stored value.
3. Performing some operation only if some variable exceeds a certain threshold.
4. Multiplying two inputs together.
5. Adding a constant to an input.

In contrast, linear operations are usually considered to include only those that can be made up from the following simple processes:

1. Adding two inputs together.
2. Multiplying an input by a constant.
3. Delaying an input by a fixed period of time.

Although these operations seem very limited, it can be shown that any nonlinear device can be accurately simulated over a finite part of its domain of possible inputs by a linear device. Such a simulation can be very cumbersome, requiring a linear device of enormous size to simulate the simplest nonlinear device, but it is always possible. A possible example of this is the human ability to perform simple arithmetic. Although human memory and processing power vastly exceed the abilities of a pocket calculator, we often depend on one for arithmetic, since this simple machine is much better at arithmetic than any human being. That we can learn arithmetic at all may be an example of linear simulation.

Another way of asserting human linearity is to say that we do not modulate our inputs, but only add to them. This is not to say that we pass input to output unmodified but for simple additions, since a constant distortion of input is not modulation as understood here. Modulation is essentially multiplication by a varying part of the signal. In practice it is usually implemented with a nonlinear element. Multiplication by a constant is not modulation since the constant is not itself a signal to be impressed on the input.

This allows us to add to our external inputs some internal input or meaning, and then to linearly transform the result. We may also linearly transform external or internal inputs before addition. Recursion (feedback) is also allowed. Altogether, our output may bear little resemblance to our external input, and

thus our role as a communications channel may be disguised. Indeed, the disguise may be so good that we forget the limitations of that role and think of ourselves as a kind of computer. But it remains true that we are not very good at ordinary computing. What is remarkable, if we accept the restriction to linearity, is that we can perform arithmetical computations at all. How do we actually do it?

2.3.4 Simulation of Nonlinear Devices

To investigate this question we can approach a typical simulation problem. Suppose that the possible inputs and outputs of a system are given in a list of input-output pairs. This list of pairs might be obtained by applying various inputs to some device and recording the output corresponding to each. If the device being investigated is a nonlinear one, then the task of developing a linear device that would be characterized by the same list of input-output pairs is the problem of developing a linear simulation of a nonlinear device. Such a simulation need only be accurate for inputs represented in the list.

A typical input-output list representing a nonlinear device is given below.

1 1

2 4

3 9

4 16

5 25

This list clearly represents a device that squares its input, provided that that input is one of those listed. Only after seeing a list of all possible input-output pairs could one say for sure if the device is indeed nonlinear or merely a linear simulation good over the given input domain.

To show that a linear simulation of this device is possible, I will give a method for developing such a device as well as a specific example of a device accomplishing this operation. There are actually many linear devices that could be represented by the above list. The easiest one to understand is the one constructed by a general simulation algorithm. The first step in this algorithm is to find a representation for the input and output. In the list above the input and output are represented by a pattern of dots on paper. Such a representation might actually be a possible choice for use in a simulation problem. It is not a difficult problem to find a linear device that transforms the pattern of dots in the number '5' into the pattern of dots in the number '25' and also does the appropriate transformations on the other input patterns. Yet this is overkill. A much simpler linear device transforms a small vector representing each input into a similar vector representing each output.

Suppose we represent all numbers, both input and output, by vectors of length 25. Each of the given inputs and outputs is represented by a (row) vector having zeroes everywhere except in the position given by the number being represented.

If input and output values are represented this way, the (rather sparse) matrix given below represents a linear operation that will transform each of the inputs into the appropriate outputs.

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Multiplying the input vectors by this matrix will produce the specified output vectors.

If the input is 5, represented as

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

the output is 25, represented as

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

This is obviously not the most economical representation of the input and output values, but it is one that guarantees a linear mapping between input and output.

It is worth noting that this device is certainly linear and as such has some properties that one would not expect of a nonlinear device. Suppose the device above is given an input not on the original list of inputs, such as

0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

The output corresponding to this input is

0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Even without knowing the matrix representation of the device it is quite possible to predict this result by applying the principle of superposition which applies to all linear devices. This principle states that the output corresponding to the sum of two inputs is the sum of the outputs that correspond to each

individual input. The input vector just given is the sum of the input vectors representing two of the inputs in the list, and therefore the corresponding output is the sum of the outputs corresponding to each of those inputs. Thus, in constructing a linear device to simulate a squarer over a narrow range of values we have produced a device that will simultaneously square more than one number in its input range.

It must be observed, however, that the success of this simulation depends upon the restriction to a narrow range. A real squarer is a nonlinear device, and according to the account given in a previous section, is therefore inherently ambiguous. If we see that the output of a squarer is 25 then we cannot know whether its input was 5 or -5. Our simulation is unambiguous: if we see the output that represents 25 we know that the input was 5.

The matrix just given for this linear simulation is obtained by a process of cross-correlation. The easiest way to understand this process is to consider the resulting matrix as the sum of a number of simpler matrices. Let each input vector and its matching output vector describe a matrix in the following way:

Each column of the matrix is a scaled copy of the input (row) vector, with the scaling values for successive columns being the values of the output vector, divided by the dot product of the input vector with itself.

Suppose the input vector is -1 0 1 2

and the output vector is 4 3 2 1

The dot product of the input vector with itself is 6, so we scale the input vector by 4/6, 3/6, 2/6, and 1/6, producing the matrix

$$\begin{matrix} -2/3 & -1/2 & -1/3 & -1/6 \\ 0 & 0 & 0 & 0 \\ 2/3 & 1/2 & 1/3 & 1/6 \\ 4/3 & 1 & 2/3 & 1/3 \end{matrix}$$

This matrix will take the input vector into the output vector.

Note that at this stage there is no connection between the shape of the input and output. The output vector may be chosen arbitrarily without consideration of the input. Indeed, the outputs of a linear device described by this matrix will differ only in magnitude, which depends on how well the actual applied input matches the input vector. This type of device can be described as a matched filter, since like other devices of that name it produces its maximum output for an input which exactly matches the intended input. A simple matched filter like this is not particularly interesting by itself, but it becomes an interesting part of a larger system when paralleled with other, similar, devices.

Suppose that the vectors chosen to represent the inputs and outputs given in an input-output list are made up so that all of the input vectors are orthogonal to one another, and all of

the output vectors are also orthogonal to each other. Such was the case in the squarer example above, as may be easily verified. Using the algorithm given above to produce a matched filter for each input-output pair will result in devices that can operate in parallel without interfering with one another.

Since the input vectors used in designing the matched filters are mutually orthogonal, each matched filter will produce its intended output for its intended input, but no output at all for any other input in the list. When the devices are operated in parallel, the inputs are applied to all devices together and their outputs added. For each of the specified inputs only one of the devices will produce an output.

The same result as could be obtained by paralleling matched filters can be achieved more economically by creating one device whose matrix representation is the sum of the matrix representation of all the paralleled filters. This gives us an algorithm for producing a single linear device which will take any of a collection of orthogonal inputs to any given outputs. This result may easily be extended to collections of vectors which are not mutually orthogonal but merely linearly independent. If a collection of linearly independent input vectors is represented as a matrix, the inverse of that matrix will be a linear operator which will transform the original collection of input vectors into a collection of mutually orthogonal vectors.

The successive application of two linear operations is itself a linear operation, so we thus have an algorithm for producing a single linear device that will transform any of a collection of linearly independent input vectors into arbitrarily chosen outputs.

It should be clear from the above brief account that the problem of simulating a nonlinear device has two subproblems:

1. Finding a suitable representation of the input and output as vectors.
2. Creating a linear operation which takes the input vectors into the corresponding output vectors.

Essentially this method is an algorithm for producing a linear device to order. The workings of the algorithm are particulary interesting as a possible model of biological systems, including the human nervous system. We may try to consider a neural net or a brain as a linear device.

2.4 Towards a Linear Associationist Model of Language and the Brain

This chapter is directed to those who find the preceding pages too abstract, and would like a more concrete model of language processing in the brain.

2.4.1 Associative Memory

The associationist school of psychology attempted to explain how the brain worked, using only a few very limited operations.

The basic principle of the associationist psychology is expressed in Proposition 10 of David Hartley's Observations on Man (Hartley 1749:65), which reads:

> Any Sensations A,B,C, Etc. by being associated with one another a sufficient Number of Times get such a Power over the corresponding Ideas a,b,c, etc. that any one of the Sensations A, when impressed alone, shall be able to excite in the Mind b,c, etc, the Ideas of the rest.

This principle was interpreted in different ways by Hartley's successors, who were concerned about what it meant to say that two sensations or ideas are associated. It was variously proposed that sensations or ideas were to be considered associated if they:

1. Occurred at the same instant (Simultaneous Association)
2. Occurred sucessively, one after another (Temporal Contiguity)

3. Occurred beside one another in the visual field (Contiguity)
4. Were noticeably similar (Similarity)

Various attempts were made to isolate one or another of these types of association as the most fundamental, and to express the others in terms of it (see Anderson and Bower 1973). In particular, most of the earlier associationists accepted only the first two types of association as distinct types and defined the other two in terms of them. Perception leading to association was either of simultaneous or (temporally) contiguous sensations. For many associationists, the most important factor in determining the relative strengths of associations was that embodied in the Law of Frequency which states a relationship between the frequency with which an association of stimuli occurs and the (resulting) strength of the association between those stimuli. Some quantitative studies of the Law of Frequency have indicated that this relationship is logarithmic, with the first few occurrences of a set of associated stimuli having a far greater effect than later repetitions.

Another factor in determining the strength of associations was the effect of related associations due to the action of some form of transitive law, which also served to create new associations out of the interaction of old ones. The existence of a scalar quantity, strength, and an interaction between associations are important to any formalization of the associationist theory in terms of vector spaces and their

associated operators.

The question of whether or not the associationist theory can be described in terms of vector spaces and linear operators was not directly addressed by the early associationists, who lacked this mathematical vocabulary. A related and possibly equivalent question was raised by John Stuart Mill in reacting to the extreme associationist views of his father, James Mill. The elder Mill had argued that complex ideas are made up of simple ideas, and in turn make up even more complex ideas, which are thus ultimately composed of the same simple ideas. His son denied that complex ideas are simply the sum of their constituents. He argued that there was a kind of "mental chemistry" which happened so that the nature of complex ideas could be entirely different than that of their simple constituents, as the nature of a chemical compound is different than it constituent elements.

John Stuart Mill thus advocated a view somewhat like the central tenet of modern Gestalt psychology, that the whole is more than the sum of its parts. The view being presented in this thesis may be treated as a denial of that tenet, and thus is more closely related to the associationism of James Mill.

In subsequent years followers of the associationist school made use of a number of mathematical methods which are only known to be accurate when the underlying mathematical model is a linear one. Methods such as factor analysis continue to be used in psychology even though a there is almost no discussion of

linear models of mental processes. The present theory could be viewed as an approach to the formalization of an associationist theory of mind using vector spaces and their associated mathematical tools.

It seems that associationist psychology can be described as including these principles:

1. Each set of associated stimuli or ideas defines an entity called an association.
2. There is a scalar quantity called strength which reflects or measures the ability of a stimulus to cause the recollection of another stimulus that has been associated with it.
3. An association will have a certain strength at any given time, but this may get stronger or weaker from time to time through reinforcement or extinction.
4. There is some interaction between associations such that the association of A and B represented by A&B, together with the association between B and C represented by B&C, produce an association between A and C, which we can represent by A&C.
5. The strength of an association A&C formed in this way from A&B and B&C will depend (in part) on the strengths of its two component associations A&B and B&C.

Briefly, suppose that we consider an association of ideas to be represented as a vector. This is a natural enough representation, since a vector is often given as a set of numerical values that are in some way associated. In order for a set of numbers to be called a vector, we have to have some

definition of length and of addition. Thus we need a scalar quantity, length, that is defined for each vector, and we need to be able to define the vector that results from the addition of any two vectors. It seems clear enough that the associationist allowed for some interaction between associations, as in points three and four above, although it is not yet clear that this interaction can be described as an operation of addition. It also remains to be seen how the strength of an association is related to the length of the vector that describes the association.

A tentative formulation of association theory in such terms may begin by identifying the strength of an association between A and B with the probability of the stimulus A evoking the memory B. On this basis we could describe the addition of associations as follows: If A evokes B and B evokes C then A can be said to evoke C.

What about the relative strengths of the associations between A and B, B and C, and A and C? Since the strength of an association is taken here to be the probabilty of one part evoking the other, then the strength of the association between A and C will be the probability of A evoking C, which can be described as the product of the probabilities of A evoking B and B evoking C. In order to translate the notion of the strength of an association into the mathematical notion of the length of a vector, and to treat the transitive evocation of a new association as a resultant formed by addition of vectors, it

would be necessary to define the length of a vector as the logarithm of the probability of one part evoking the other, so that the product of probabilites will become the addition of vectors. This is quite compatible with the familiar psychological principle that there is a logarithmic relationship between stimuli and perception.

This description can provide the basis for a mathematical model of associative memory, but it is not yet sufficient for a physical model. In the next section I review some simple physical models of memory. The significance of these models for language study lies in differences between models that may have reflections in the properties of language.

2.4.2 An Optical Model of Associative Memory

A concrete example of a linear device that can serve as a model for some human mental processes is provided by newly discovered techniques of holography. Physicists apply the term holography to a process of lensless three-dimensional photography invented by Dennis Gabor. (Gabor 1949)

The best known use of the hologram model of a brain function was by the American psychologist Karl H. Pribram (Pribram 1971). Pribram ws interested in the location of stored information in the human brain. Drawing upon experimental evidence that any given item of memory can be recovered in spite of almost arbitrary brain lesions, Pribram argued that information must be stored diffusely throughout the brain, as

information is stored diffusely over the surface of the film in one of Dr. Gabor's holograms. However much this may seem a mere metaphor, Pribram argues, it could be a strictly accurate, literal statement, since a hologram is a spatial fourier transform of the light leaving the suface of an object. Pribram suggested that the word hologram just means spatial fourier transform. He cited a certain amount of physiological evidence that the brain performs the mathematical operation of taking the fourier transform of incoming data.

Pribram's reasons for wishing to treat memory storage as a holographic record depended upon the many remarkable properties of holograms. These are in fact just properties of the fourier transform, familiar to mathematicians for decades. What is remarkable about Dr. Gabor's process is the ability to perform this complicated mathematical operation (in two or three dimensions) by a simple optical process.

2.4.3 Holograms and Photographs

Let us compare the two processes by which information about the visual appearance of an object may be recorded on film. One of these is photography; the other holography. In making a photograph, a lens is used to create an image of the object on the film. In making a holograph, light reflected from the object, together with light from a reference beam forms a pattern on the film as waves of different frequency and phase interfere with one another. The piece of film can be placed so

as to capture almost any part of this interference pattern. The developed piece of film will diffract light from a suitable source to reproduce a copy of the original reflected light from the object. [6]

For the purpose of this comparison it is interesting to ask what determines how the photographic or holographic film will darken at a specific point. In the case of the photograph, each point on the film corresponds to a specific point on the object. The darkening of the film at a point depends on the amount of light reflected in the direction of the lens by the corresponding point on the object. For holographs, the case is different. The darkening of the film at any point depends in a complicated way on the light reflected from all points. Information about the appearance of the object is stored diffusely over the entire surface of the film. Any small piece of the film will suffice for reconstructing light patterns from the entire object, although these may give a view of reduced quality or scope.

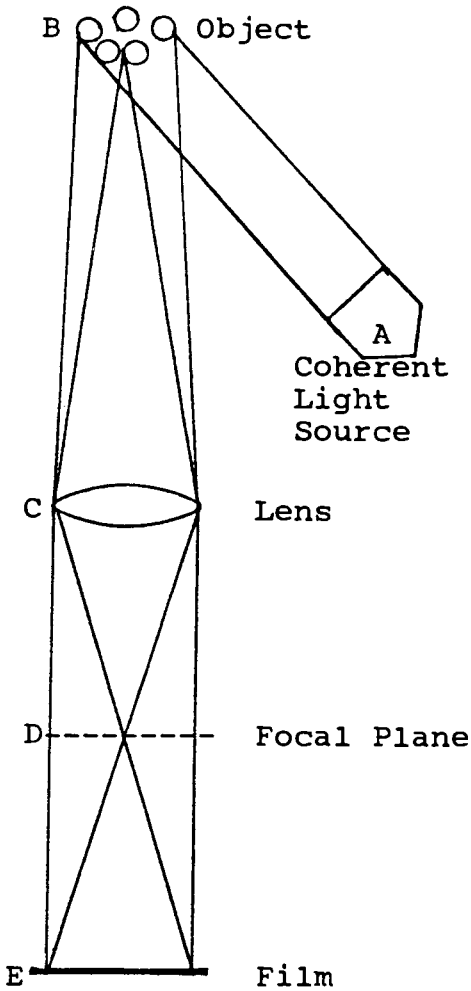
Figure 13 shows a simple process for recording information on film.

At A is a coherent light source, which illuminates an object, B. The focussing effect of the lens at C casts an image of the object on a piece of film at E. If the film is exactly as far from the focal plane of the lens, D, as the lens is, then the image on the film will be quite clear. It will be an

---

[6]To understand the role of the reference beam in this process, see the next section on the associative property of holograms.

Figure 13. Recording Information on Film



The same apparatus can be used to make either photographs or holograms. To make a photograph, the film is placed at a distance from the focal plane of the lens. As the film is moved closer to the focal plane, the image gets more and more 'out of focus' until it is unrecognizable. If coherent light is used, a Fourier hologram can be made by placing the film on the focal plane.

inverted two-dimensional view of the object. Suppose the film is now moved a small distance towards the focal plane of the lens. The image will become more and more blurred as it is moved closer to the focal plane.

To understand the effect of blurring, consider what happens when the film is moved all the way from its original position to the focal plane of the lens. For simplicity, assume that the object is two-dimensional. In this case the pattern of light on the film at the focal plane is the fourier transform of the pattern on the surface of the object. Another way of describing this is to say that by moving the film to the focal plane of the lens we have made a hologram of the object. Thus we could construct either a photograph or a hologram using the same experimental situation, depending only on where the film was placed. In fact, if the film is placed anywhere between the two extremes, a usable record of the original transparency will result.

The image produced by placing the film anywhere except at the focal plane will be somewhat blurred; it will be literally "out of focus". But, an out-of-focus photograph is still as complete a record of information as the original photograph. In fact it may be more complete. Since a hologram of a three-dimensional object includes enough information to reconstruct a three dimensional image, it will include more information than an ordinary two-dimensional photograph of such an object. A blurred photograph may be considered as being half

way between a photograph and a hologram, and may include an intermediate amount of information.

Ordinarily lenses are not used in making holograms, in which case the relationship between object and hologram will not be exactly describable by the fourier transform, but will be one of a family of related integral transforms. Reconstructing such a hologram involves inverting whatever integral transform has applied, and this can be difficult unless the associative property of holograms is used.

2.4.4 The Associative Property of Holograms

A hologram that is made in the above way as a record of a collection of adjacent objects rather than a single object, may be illuminated with coherent light reflected from a single one of the objects after the others have been removed. If this is done, the remainder of the collection of objects will be seen. The resulting hologram may be considered an associative record of the collection of objects, since light reflected from any object in the collection may be used to reconstruct the image of the others. What is most interesting is that multiple exposures using various collections of objects can be created and reconstructed without much interaction between them. In other words, the light from a single object from any collection can be used to reconstruct the images of the other objects, with which it was associated.

The piece of film used to make a hologram may be double exposed, or even exposed several times. At each exposure a collection of objects may be recorded. If none of the collections have any objects in common,[7] then light from any object will produce a perfect view of exactly those objects holographed in the same exposure. But if a single object occurs in two or more collections, the light from this object would reconstruct the images of all of these collections simultaneously, and these images would be superimposed over one another. A single exposure can be viewed as the construction of a matched filter, and then the additional exposures add other matched filters to be used in parallel.

This phenomenon has been described many times, and various authors have proposed exploiting it to create a content-addressed memory or associative memory for use in a computer. Various experimental versions of such an associative memory have been devised and tested. The existence of associative memory models based on holograms is important to the present theory, since the fourier transform is known to be a linear operation. One problem with holographic models of memory is that although the experimental setup produces a pattern of light on the film which is related to the pattern of light at the surface of the object by a linear transformation, the actual recording of this pattern on the film by chemical means is not a linear operation.

---

[7] or, more precisely, if their individual holograms would be orthogonal to each other,

This problem can be overcome through the use of an approximately linear storage medium. An example of such a medium is photopositive film, which darkens if deprived of light, and then bleaches to a lighter colour when light is applied. This medium can be driven back and forth between dark and light shades by slowly varying illumination. Its colour at any moment is some weighted average of the light intensities over a preceding period of time.

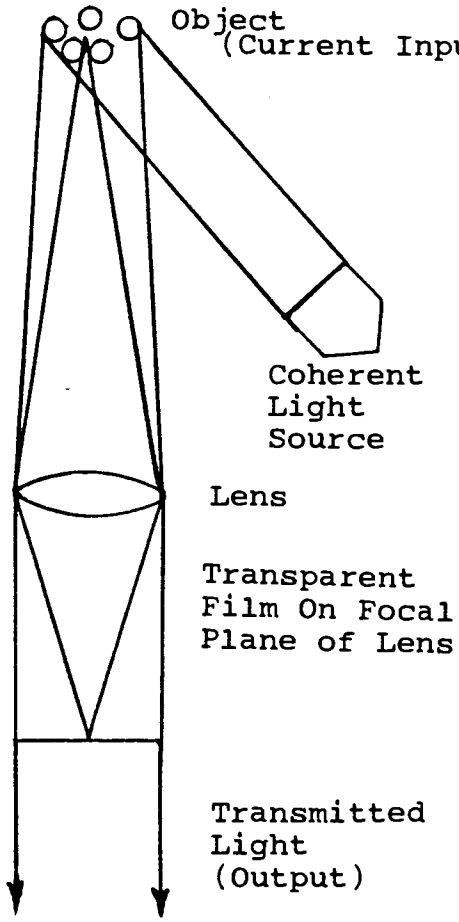
Figure 14 shows a simple model of associative memory using photopositive film as the storage medium. In this model, light passing through the system forms a hologram of objects appearing on the left. This hologram also serves as filter, determining what image will be seen, by modulating the same beam of light that affects the film. Points on the film that regularly receive a lot of light will be bleached by it, and thus pass light more easily. In this way the model is similar to some proposed models of the human nervous system, in which the response of a synapse is supposed to be affected by the number of impulses crossing it. This model, like those proposed by Kohonen and others[8] is not actually a linear device, since the film modulates the same beam of light which changes its colour, and this modulation is a multiplicative process in which the current light intensity is attenuated to some fraction of its value by the film.

Since I argue that human beings are actually linear devices, I do not think this model is quite correct, but it is

---

[8] Kohonen (1977), described below

Figure 14. Simultaneous Storage and Retrieval in an Optical Memory System



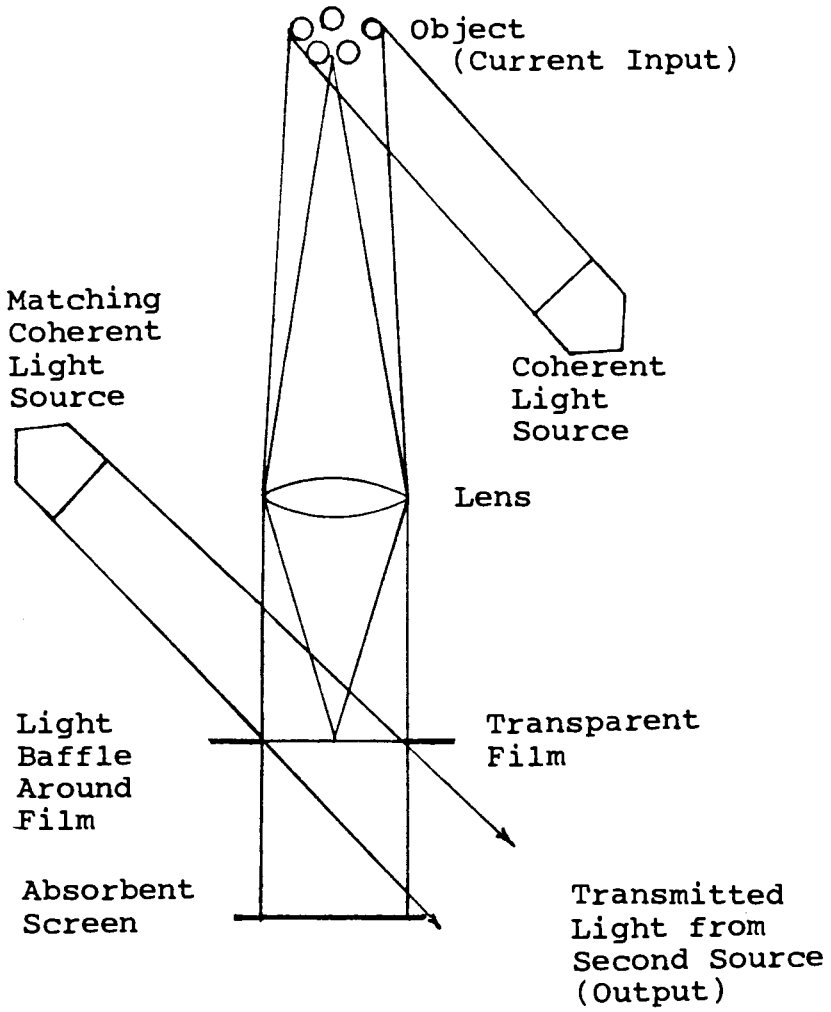
A model of an associative memory can be made using a film of a special material that slowly changes colour on exposure to light. The film can be exposed to light from several objects, and the holograms of those objects will be superimposed. Light from the current object is filtered by the film, but may still form a real image. The contents of this image will depend on the similarity between the current object and objects previously recorded.

of heuristic importance, in that it shares many features of a truly linear model. In particular, the response of this device to sequences of stimuli presented to it very quickly are of interest. If we supply the device with light patterns that vary too fast for the film to respond by changing colour, then we can use the device for associative recall without concern for the nonlinear nature of storage-recall sequence. The rate of colour change (which depends partially on the average intensity of illumination) determines the frequency response of the device. For very low frequency (or very high intensity) changes in input the device will be nonlinear, but display both memory storage and recall. For higher frequencies the device will be approximately linear, but useful only for recall.

The properties of devices that are approximately linear over some range of inputs are familiar as practical communications systems. The linearity of such devices is often improved by negative feedback, and this suggests a possible way of improving this model.

Feedback also serves a critical role in another possible model of memory that should be mentioned at this point. The problem with the above model is the way stored data modulates the current flow of data, which is a nonlinear process. But stored data can be used in another way: suppose the incoming data encoded as light impinges upon the photopositive film along one path, while the same film passes another beam of light travelling along another path,as illustrated in Figure 15. If

Figure 15. Possible Linear Version of Optical Memory



If two separate light sources are used, a less useful but more accurate model may result. This version is more likely to be linear, since the current input will not modulate itself, but without the addition of external feedback it is little more than a low-pass filter.

the second beam of light is a constant, then it may be modulated by the stored data on the film, and passed to the output. Such a device is linear, since the multiplicative process of modulating a constant beam is a linear operation. The problem with this kind of device is that the output is only a weighted average of recent inputs, and not particularly useful. Although data is stored, and stored data affects the output, so that the device is some kind of memory, it is not much more than a low-pass filter, and thus not very useful. As I have tried to show in earlier sections, the addition of positive feedback to a device of this type can result in something more interesting and useful.

To sum up, an associative memory can be implemented in with a nonlinear optical device, which negative feedback could make more nearly linear, and a simple filter with some of the properties of memory can be implemented as a linear optical device, which positive feedback can make more useful. To implement a model of the brain one need not be restricted to one or the other of these methods, but may use a complicated mixture of them both. What they have in common is the use of feedback. It seems that some form of recursion or feedback is necessary for any adequate model of the brain.

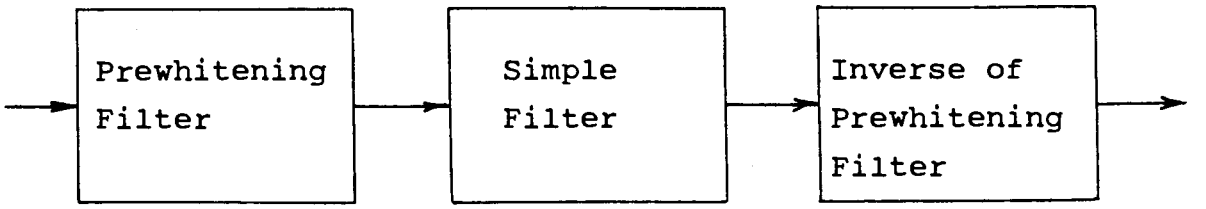
Kohonen's study of associative memory (Kohonen 1977) includes such a suggestion, and a linear model of the cerebellum has been made by David Marr and James Albus (Albus 1975). These models of neural networks as linear devices are limited in one

fundamental way. They are linear as regards information retrieval, but the storage of new information involves modifiable parameters, and if we consider that incoming information serves to modify these parameters, the result is actually a nonlinear device.

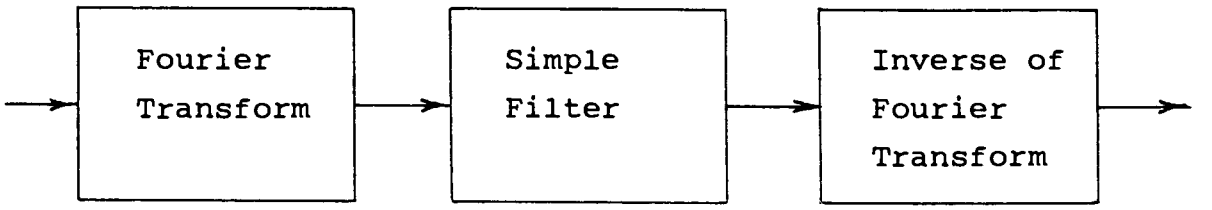
The models which store information in modifiable weighting coefficients are technically not linear models, but multilinear models. A multilinear device is one with several inputs, which acts like a linear device with respect to some of those inputs if the others are held constant, but is nonlinear if all inputs are considered together.

All of these multilinear models incorporate a step that produces weighted sums of input vectors. This is the most basic type of linear operation, and is usually represented as a matrix multiplication. But linear operators are known that cannot be described as producing weighted sums of their inputs. Recursively defined operators, which incorporate feedback loops, have what is described as an infinite impulse response, and so cannot be represented as multiplication by any finite matrix. Norbert Weiner(1949) showed that such operators can be described using a finite matrix, if the operation of matrix multiplication is combined with the fourier transform and its inverse. A Wiener filter consists of a nonrecursive (finite impulse response) linear operator 'sandwiched' between two integral transforms.

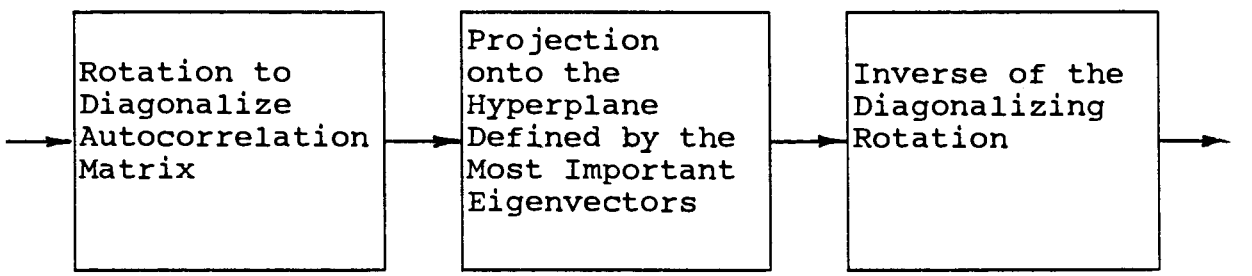
If we represent a linear device as a Wiener filter, as in Figure 16 we can see its relationship to the holographic models



a) Weiner Filter



b) Frequency-Domain Convolution



c) Pseudo-Identity Filter

Figure 16. Use of Linear Transformations in Implementing Linear Devices

discussed previously. Pribram's view of memory involves storage of information after a fourier transform, and is similar to Kohonen's model, in which the operation following the fourier transform is redefined as new information is stored. If a Weiner filter is treated as a realistic model of the human brain, information storage is more likely to be in the transform step, since integral transforms such as the fourier transform involve memory.

The models mentioned are models of vision, in which the two (or three) dimensional visual field is transformed. The transform applied is a spatial fourier transform, applied at a particular instant of time. For speech, only the time dimension is important, so we may imagine a Wiener filter for speech, involving a fourier transform, simple filtering, and an inverse transform. It would be quite redundant to add to this model a memory-storage system that adjusts filter coefficients or weights, since the temporal fourier transorm and its inverse must both involve information storage. A three (or four) dimensional model of vision involving a spatio-temporal fourier transform would similarly need no memory-storage system. When we omit the operation of adjusting filter weights, we omit the only nonlinearity remaining in the model. A pure linear device remains, and that is a most attractive model of the brain.

It is true that on the minute physiological level memory may be a nonlinear operation, but this nonlinear operation may be hidden within some larger linear operation. Suppose we

attempt to simulate a Wiener filter on a digital computer. The computer will have to act as a device that performs the linear operation of taking the fourier transform of its input. To do so will involve the use of computer memory elements, which are nonlinear. But the nonlinearity of individual memory elements is hidden within the simulated linear device. The theory being suggested here is that a human brain considered as a complete system is perfectly linear, but I will admit that memory storage at the microscopic level is nonlinear. It is important to remember that making an network of nonlinear devices may result in a linear device, as nonlinearities cancel out, but a network of linear devices can only be a linear device.

The models of Kohonen and others may not be entirely linear in operation, but they are the closest yet developed to a linear model of the brain. In operation these models behave as associative storage and recall systems, with some processing ability. Their discovery followed attempts to model the operations of neurons.

It is common to model the neuron as a device that passes its inputs to its output as a weighted sum, with variable weights. Variable weighting would be a nonlinear operation. I propose instead that neurons are also constrained to the performance of linear operations, for the same reason the brain as a whole is: linear devices are better for the transmission of information.

The algorithm given above for creating linear devices to order can be studied to see how neurons may combine to perform complex filters capable of difficult transformations such as those involved in human language processing. This algorithm can be described in several different ways, but it is useful to consider it as an application of a direct sum decomposition. Any linear device can be expressed as the sum of a number of other linear devices, usually simpler ones. The decomposition of a linear operator into simpler linear operators is known as a direct sum decomposition. The particular decomposition that is most interesting for the present purposes is the decomposition into simple matched filters.

A matched filter is any filter which is matched to a particular input, so that it produces its maximum output when given that input, and as little output as possible for any other input. The exact form of the output can vary between one type of matched filter and another. Indeed, the output can have any desired form.

Most biological systems are made up of a number of separate components which work in parallel. Many modern biologists accept the idea advocated by Sir Peter Medawar that the responses of simple biological units is elective rather than instructive, or in other words that the simplest life forms have a number of simple responses which they can choose between, but they do not learn new respones. As a simplification of this theory we may imagine biological units that have just two responses, a null

response, and any other response. Such a unit can be considered as a matched filter, as described above.

The advantage of considering simple biological units as matched filters is that composite life forms containing many units operating in parallel may be then considered as a direct sum of matched filters. This reduces the question of the linearity of biological systems to questions about the linearity of their components and about the way in which these components combine.

For example, we may consider the question of the linearity of the human nervous system. It has long been noted that neurons have an all-or-none response. This can certainly be described as nonlinearity, but there is no advantage to be gained by this description if a linear one is possible. One way of making neural responses look more linear is to consider the output of neurons in a different way, not as an amplitude but as a frequency. From a mathematical point of view this is a change in the underlying field over which the space of responses is described. Experimental stimulation of peripheral nerves which respond to inputs of varying amplitudes with outputs of varying discharge frequencies shows a correlation between input amplitude and discharge frequency that is approximately logarithmic over most of its range. Again a change in underlying field can help: the input space can be defined over the field of logarithms of actual input amplitudes.

Within the brain the inputs of neurons include the synaptic connections with other neurons, and here the underlying field of the input space of a neuron may be conveniently taken as the same field of frequencies as the output space.

All of this is to suggest that individual neurons be modelled as linear devices, particularly as matched filters or sums of small numbers of matched filters. The task of producing such a model may be difficult by comparison with the task of isolating the familiar all-or-none response, but the advantages are considerable. We may incorporate the familiar response patterns of neurons into linear operations if we redefine the underlying linear space. Linear spaces are defined in terms of an underlying field, and by giving a set of orthogonal basis vectors, such as the X,Y, and Z axes used in describing three dimensional space. Linear spaces may be of infinite dimension, in which case the term 'basis function' may be used instead of basis vector, and any set of orthogonal functions may be used as a basis for a linear space.

In modelling neurons, we could use functions representing the outputs of neurons firing at different rates. Each function would be an infinite series of spikes, regularly spaced, with the number of spikes in a given length of time the critical difference between them. In considering individual neurons, we would be interested in the expected firing rate or mean firing rate for each neuron in its usual environment. The expected

firing rate could be a complex weighted sum[9] of the inputs, to which is added a complex weighted sum of past firing rates, multiplied by a (time-varying) constant. This addition of a complex weighted sum can take the place of the multiplicative operation found in most models of neurons if the environment provides enough feedback of a suitable kind. The problem is identical to the second optical model discussed above. It is also similar to the linear Markov model of human language processes.

The total model of a human brain can thus be similar to the model of each of its neurons. In implementing a linear Markov filter we can perform a single counting operation on a text sample, or we can count continuously. The first approach creates a constant linear device that 'knows' some part of the language, while the second would learn the language. In modelling an individual neuron we use firing rate, or firing probability; in modelling a whole language processing system we treat uttering any symbol as a kind of 'firing' and need to establish the probability of that occurence. This probability is just the number of times it has already occurred, divided by the length to time the device has been in operation. The former is a sum of past input occurrences, and the latter a 'time-varying constant'. Thus the device can be considered as a linear filter, and it serves as a model of human language learning as well as a model of language processing.

---

[9]using time-varying but 'constant' weights: constant with respect to the input signal

Both of these models show a kind of learning: neurons acquire a mean firing rate, and the language learning filters acquire mean frequencies for uttering particular expressions. As statistical linguistics has demonstrated, these frequencies are remarkably stable and characteristic of the individual and his history. and this learning process is essentially the addition of probabilities. As indicated above, it is also similar to a linear model of the neuron. I believe that the best models of single neurons, complex networks of neurons, and the whole brain is a linear one that does include associative abilities, and that many of the properties of human language support such a model.

The arguments which have been used to support the hologram model of memory point out how all memories seem to be stored in all parts of the brain at once. This is in sharp contrast with the usual practice in digital computers where each data item has a specific memory location. The distributed nature of the human memory system has suggested models of the brain based on the Fourier Transform. But any other integral transform would be just as valid a conclusion, since any integral transform depends on integration over the whole (spatial) domain. Indeed, none of the evidence proves that all memories are stored equally at each and every memory location, only that memory seems to be widely distributed. Thus, we are only justified in concluding that the brain is a device of very high order. I have tried to show that devices of high order are sufficiently powerful to carry out all

of the operations we know the brain to be capable of, even if of very low degree, or even linear. Linear devices are the most interesting case, since they have unique advantages in message encoding, transmission, correction, and decoding.

PART THREE

Linguistic Consequences and Conclusions

3.1 Results for Various Languages

3.1.1 Evaluation of Coordinate Assignments

The problem of evaluating coordinate assignments is complicated by the existence of several different methods which do not produce exactly identical results.

In each of the scatter diagrams that follow the symbols appear in loose clusters. Symbols representing vowels tend to appear on one side of the diagram, with a wide spacing between them. Although it is quite clear that this pattern holds in each of the languages studied so far, the analysis of data that falls into such clusters is not entirely straightforward. Another recognizable cluster in each of the languages being discussed is the cluster of labial stops and fricatives which occurs on another side of the diagram. This cluster is a much more tightly knit one than the cluster of vowels.

In these diagrams two sounds appear close together if they are likely to appear in similar context, or, in other words, if they are functionally interchangable. The fact that the vowel sounds form a group or cluster indicates that the vowel sounds are all more or less interchangable. Similarly, the labial stops and fricatives are all more or less interchangable. But the vowels are quite different from the labials. Not only are the individual vowels very easily distinguished from the individual

labials, but the cluster of vowels does not resemble the cluster of labials. If a diagram such as those below is given with missing or incorrect labels for the points, it would be unlikely that the cluster of vowels would be confused with the cluster of labials, or any other recognizable cluster. Thus, although the different languages produce different patterns of sounds, the patterns are similar in ways that make types of sounds recognizable.

The patterns of sounds given by the above methods do not depend on any prior information about the sounds of the symbols, rather they are a way of extracting such information. Given an entirely unknown system of symbols for representing speech sounds, it seems likely that the symbols could be at least partially decoded by representing them on a diagram as before. Those symbols that appear in a widely spaced cluster on one side of the diagram would probably be vowels, those that appeared in a more compact cluster on another side of the diagram would probably be labial consonants. Of the vowels, it may be noted, in examining the above diagrams that the front vowels tend to be on the side of the cluster closest to the labial consonants. In this way the various unknown symbols may be partially deciphered.

It is not clear at the present time how successful this method can be, nor is it clear whether it depends on the linguist/interpreter's insight, or can be entirely automated. I suggest that it is a relatively simple pattern recognition and

matching problem that can be solved by computer. Only tests with a lot of data from exotic languages will help to decide this matter.

The available data from a small number of languages shows that the patterns of symbols in these languages are very similar. If this holds true of the remainder of human languages, then the system of relative coordinates that have been developed for each language can be compounded into a system of absolute coordinates. This may be accomplished with the aid of methods from pattern recognition theory. It may be found that all languages have a distinctive cluster of vowels and of labial consonants. The best coordinates for each language are, it seems, ones that pass through the centre of each of these clusters. If each language is treated in this way, similar patterns may be created, implying that the coordinate systems arrived at for each language are basically similar. It only then remains to treat the coordinate values arrived at for each language as data, allowing comparison of languages. Languages with differing coordinate systems can be compared if there is some reason to believe that the coordinate systems are basically similar, and if there are enough different languages to provide a large scattering of data.

3.1.2 The Interpretation of Scatter Diagrams

It must be noted that the scatter diagrams discussed in this section are included as examples only. The research reported here was directed at the problem of finding algorithms and testing them, not at producing usable results. Thus linguists are advised that the results given are based on limited text samples and may not be reliable.

The comments given here are based on the notion that the actual positions in a higher-dimensional space are represented here in very abbreviated form, but that the two dimensions given are almost optimum for the problem.

The text samples used in two of the examples were continuous text of a moderate length, constituting the first chapter of the book of Mark in the New Testament. The other examples are based on word lists. Lists were compiled of very common words in four languages, and because of the commonness of these words, I believe these lists are more representative of the language than continuous text samples of the same size. One scatter diagram represents the results of applying the program described above to all of the four word lists taken together. This procedure gives a better picture of the interrelationship of speech sounds, since eccentricities or deficiencies in the use of alphabetical symbols in one of the languages can be made up by another. This diagram (see Figure 17) shows several features which are harder to see in the other diagrams.

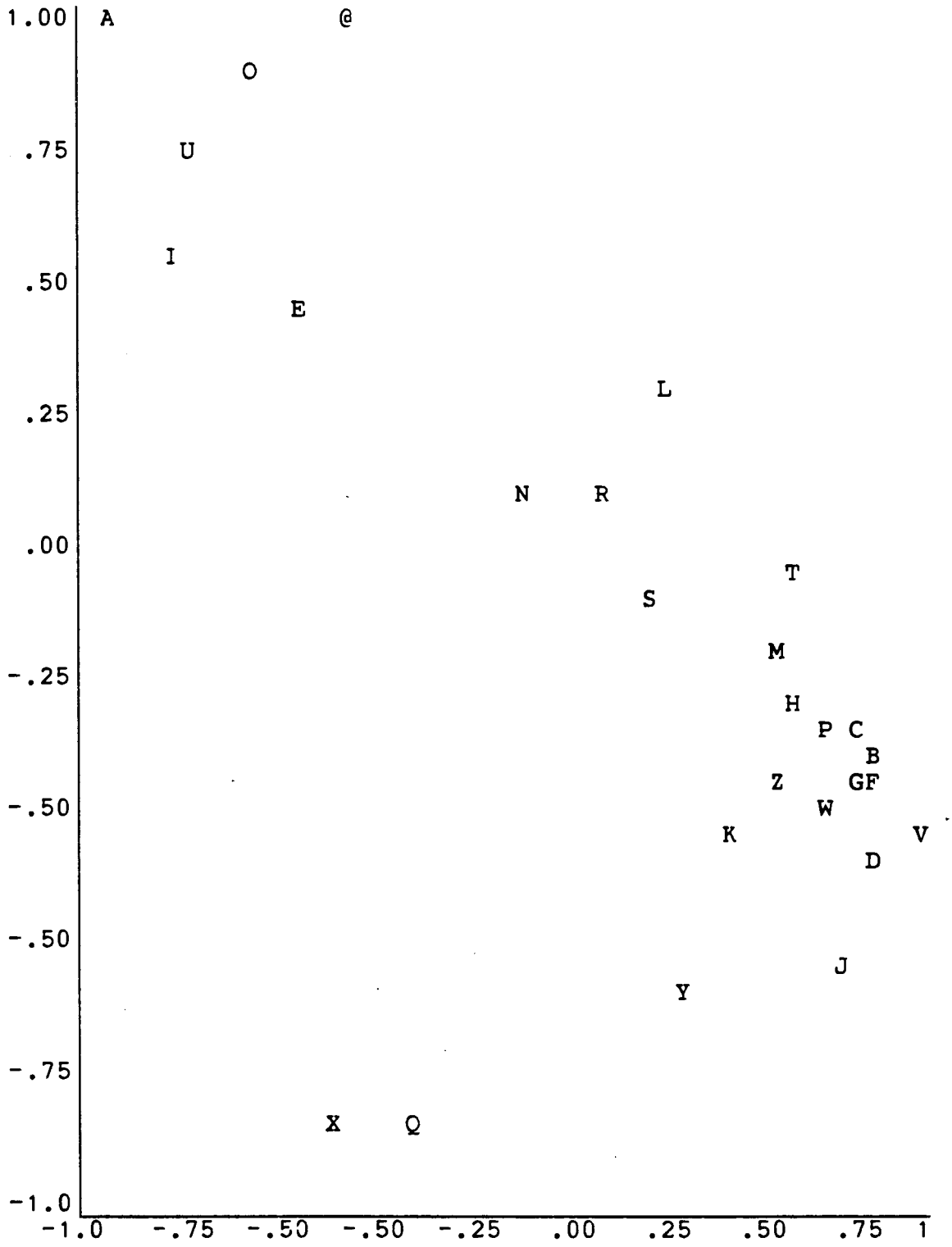


Figure 17. Analysis of Word Lists from Four Languages

As in all of the following diagrams, the vowels are clearly separate from the rest of the letters. With the vowels occurs a symbol '@' representing the space between words, which is indistinguishable from a vowel in all of these diagrams, and indeed in all other produced in the course of this research. This may suggest that the role of vowels in these languages is as transition elements between components, rather than as the core or center of a syllable as often believed.

Near the vowels occur three letters representing continuents or semivowels: 'n', 'r', and 'l'. This treatment of 'n' as a semivowel is typical of the results obtained. In none of these experiments was 'n' placed near to 'm' as if forming a class of nasals. Instead, 'n' was either near the vowels or the coronal consonants 't' and 'd'. This does not mean that there is no natural class of nasals in these languages, but only that the importance of this classfication is less than some other divisions of the letters.

The position of 'y' is interesting. This symbol rarely appears as a vowel or semivowel, but is treated as a consonant, although usually at a distance from the central core of consonants. The 'j' symbol occurs near 'y' and near 'd' but otherwise away from the mass of ordinary consonants. This is appropriate for a letter that sometimes represents an affricate (the only single letter to do so in English, where it is an affricated /d/.)

Amongst the class of ordinary stops and fricatives, we can see a difference between the voiced and voiceless versions of each sound. If we look at the pairs of letters p-b, f-v, t-d, s-z, and g-k, we can see that in every case the voicless sounds, (p,t,k,f,s), appear more towards the left of the diagram than their voiced equivalents (b,d,g,v,z). In all of these but the pair k-g, the voicless version is also towards the top of the page. The deviance of k-g may be due to the inadequacies of the text sample. The letter 'k' did not appear at all in some languages, and only rarely in the rest.

We may now look at the diagrams for individual language with more understanding of what to look for. The first two diagrams to examine are English and German, which show their family relationship, as do the diagrams for French and Spanish which follow. The diagram for English (see Figure 18)on the following page is similar to the diagram just described for all four languages, but the consonants are not clustered so closely together. Here we can notice a tendancy to distinguish between the coronal consonants (t,d,s,n, and perhaps r) and the noncoronal class containing labial and velar consonants. In every case except 'k' the labial and velar consonants are further to the right than the coronal ones. Again 'k' is poorly represented in the sample. In this diagram the distinction between voiced and voiceless is not visible. This may be interpreted as meaning that the distinction between coronal and noncoronal sounds is more important than the distinction between

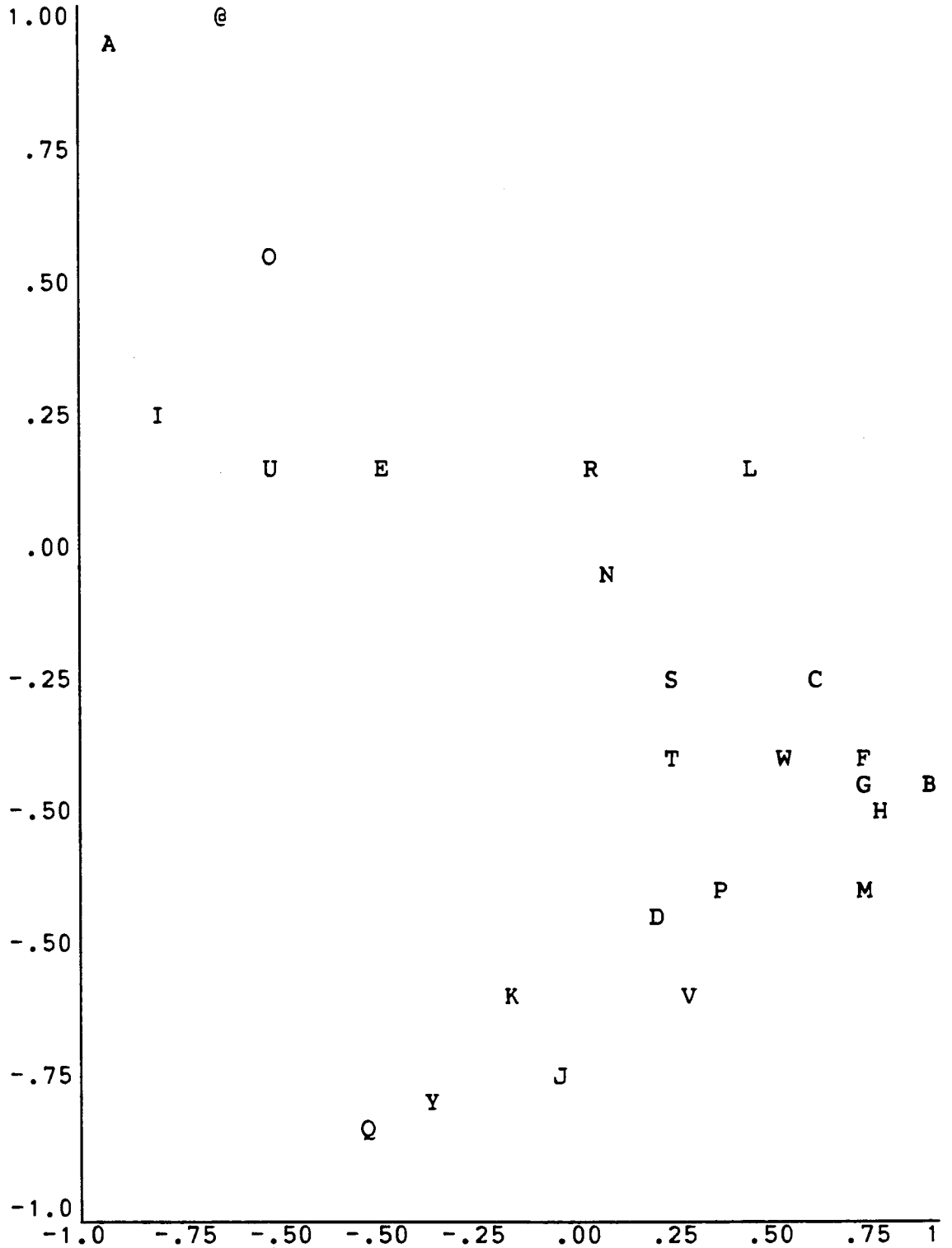


Figure 18. Analysis of English Word List

voiced and voiceless sounds. Considering the tendancy for voicing to change in context, this conclusion is possible. Thus 't' is sometimes pronounced more as a /d/ intervocalically, and 's' if often pronounced /z/, but 'p' or 'k' would rarely be pronounced as a /t/ and 'b' or 'g' rarely as a 'd'.

For German (see Figure 19) the relative importance of these two distinctions is reversed. In every case the voiceless consonant is higher on the page than its voiced equivalent. 'Z' is a rare letter in English and did not appear in the sample. In German it is much more common, and occupies a place below (and slightly to the right of 's'.) The right and left dimension does not seem to represent any reasonable sound distinction in the consonants, although there are more labial and velar consonants to the right, and more coronal consonants in the middle. For German the distinction between consonants articulated in different places may be just as important as in English, but this particular projection does not reveal it. One reason for this is a need to use the left and right dimension for distinguishing between consonants and vowels. In the English example the vowels were much higher on the page than the consonants. Here the difference between consonant and vowel is more clearly one of horizontal position, as if the diagram had been rotated slightly counterclockwise.

It is interesting to note that 'n' appears in the cluster of vowels in German, and is much further from 'r' and 'l'. It is true that nasal vowels are much more common in German than in

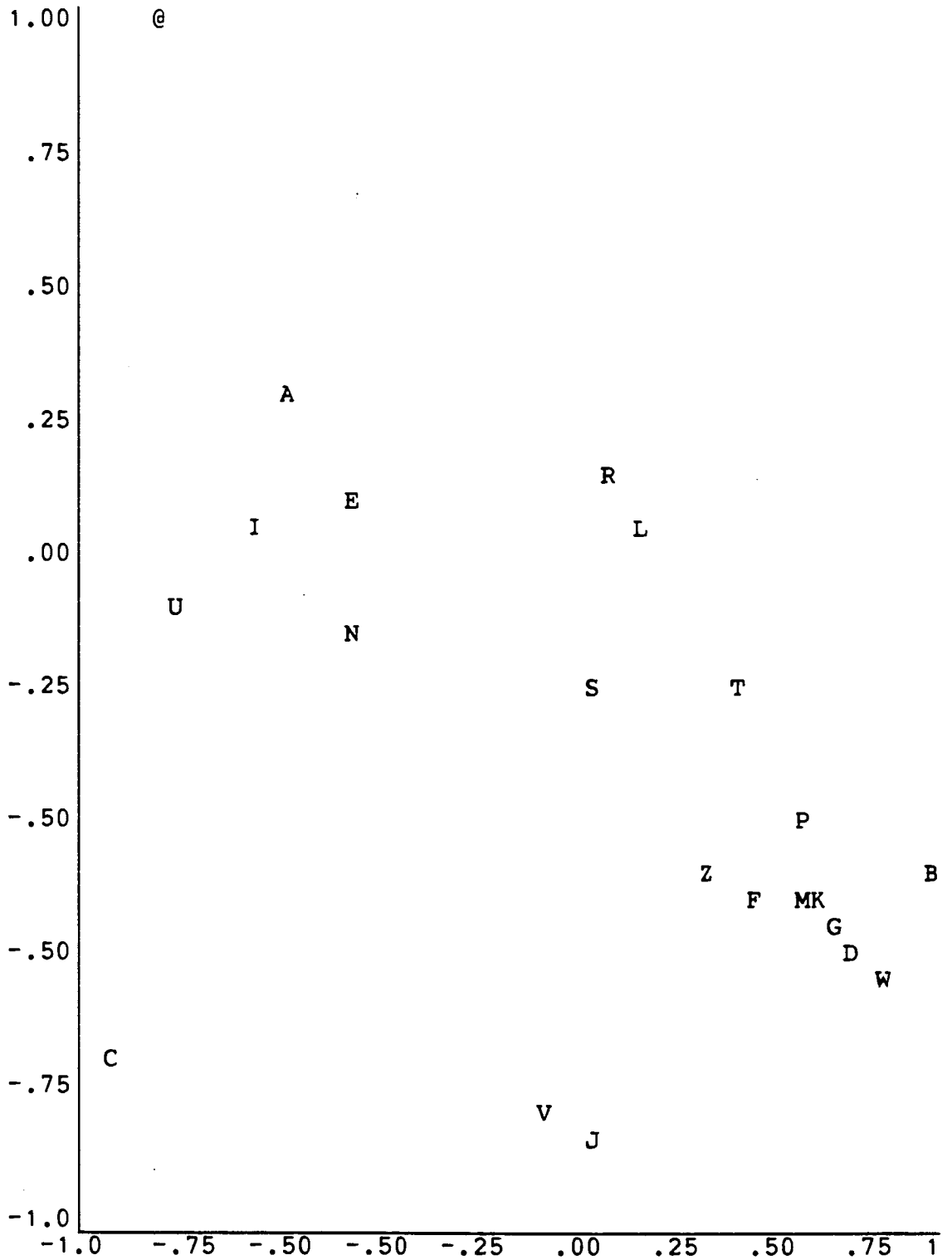


Figure 19. Analysis of German Word List

most dialects of English, but this may or may not be an explanation.

The position of the 'c' is odd, and hard to explain, although it may relate to the frequency of the 'sch' cluster in German. This suggests that the principal role of 'c' is as part of a fricative. We may notice that the large empty space in the lower left hand corner of the diagram is bordered by 'c', 's', 'z', 'f', and 'v', (moving clockwise from 'c') and that these are exactly the fricatives.

We may compare the diagram of French (see Figure 20) with English and with the Spanish diagram that follows it. French shows similarities to both English and Spanish, but is quite different from German. One similarity that can be noted to the German case is the position of the 'n', much closer to the vowels than any other consonant, and indeed closer to the vowels than the consonants. The pattern of sounds in French is very nearly one-dimensional, with the various sounds forming a snake-like curve with little width at any point. Because of this effect, few of the familiar linguistic distinctions other than vowel and consonant are apparent. In French we do find 'm' getting closer to 'n', and the letters 'n', 'r', 'm', and 'l' almost form a class, although 'm' is also close to 'p', 'b' and 'f'. Oddly enough the letters 'c' and 's' are near 'b' and 'p'. This may also be seen in the Spanish example below. This is actually an anomally of the projection, as can be seen from a different Spanish example discussed later. The overall

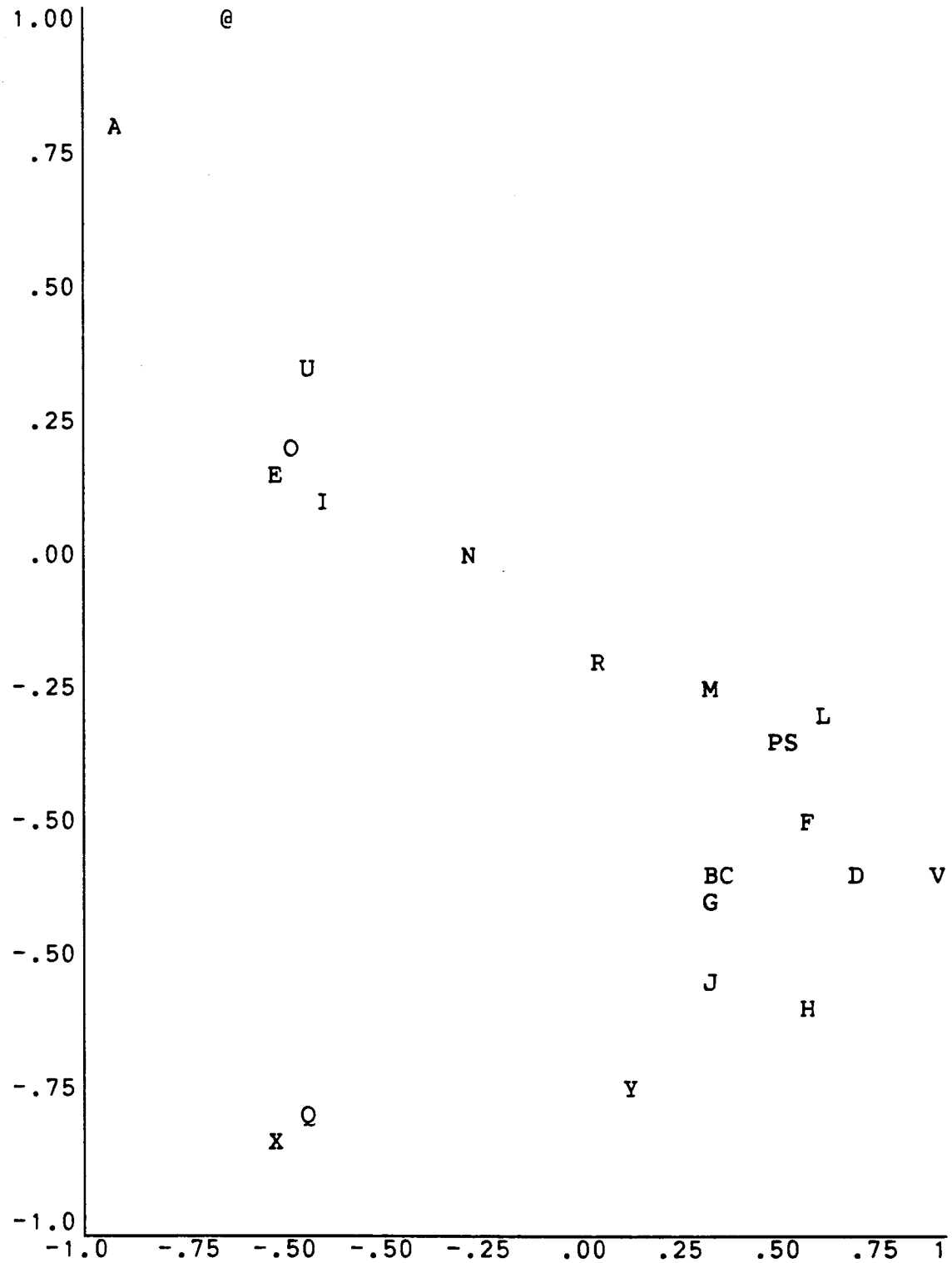


Figure 20. Analysis of French Word List

similarities between the French and the Spanish examples (see Figure 22) are striking. In the Spanish case the curve is curled around a litte more, so that the sequence of vowels 'a', 'e', 'o' forms a right angle to the sequence 'o', 'u', and 'i'. Whether or not this reflects some significant distinction between Spanish vowels is a difficult question. It may be that the proper description of Spanish is as a one dimensional curve. This notion is born out by comparing English and Spanish using a different algorithm and a better text sample.

The following two diagrams were prepared using factor analysis based on extraction of eigenvectors. The problem with ordinary factor analysis is that it produces a set of coordinates 'custom-fitted' to the data, so that two different sample of data from two different languages will each have their own set of coordinates and not be very comparable. Nevertheless, the comparison of English and Spanish is striking.

In the Spanish example (see Figure 20) the optimum set of coordinates leaves consonants at the top, with little of linguistic interest in their distribution, although the labial 'p', 'b', and 'm' are neighbours, with voiced consonants more to the right of the page than voiceless ones. After that, the pattern of letters extends around in a semicircle, that is clearly one-dimensional. Indeed, we have here a fairly simple pattern, since the pattern of consonants could be described as voiced clockwise from voiceless (for the usual pairs, except perhaps for the pair q-g of velar stops,) and the vowels being

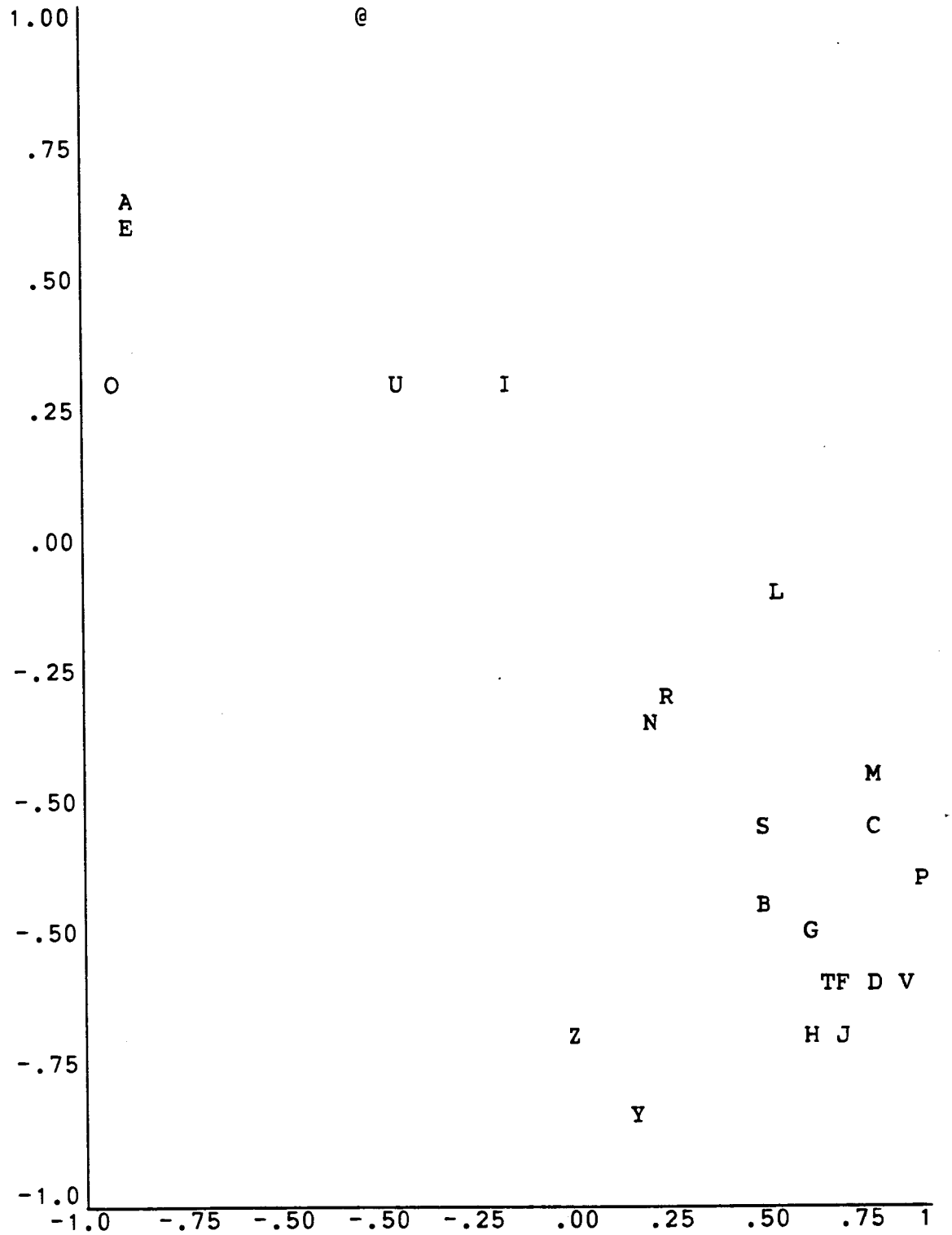


Figure 21. Analysis of Spanish Word List

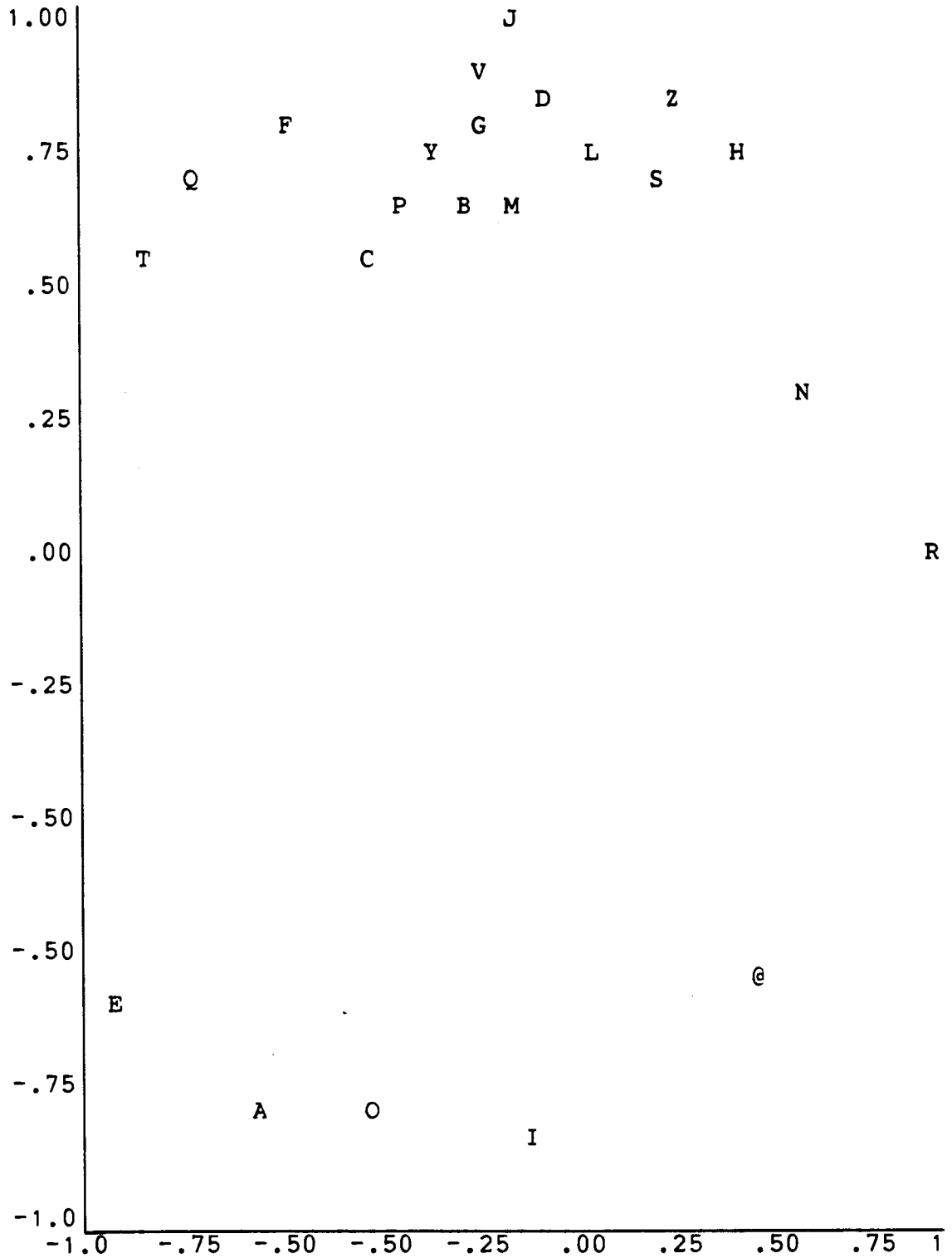


Figure 22. Analysis of Spanish Text Sample

more voiced than the consonants just continues the pattern.

This one-dimensional pattern is in sharp contrast to the English case, which does show a slight tendancy to a circular sweep, but it is clearly two dimensional. Note that the 'customizing' process of factor analysis makes details of position incomparable. Thus it does not make sense to wonder why the English diagram (see Figure 23) seems rotated slightly counterclockwise from the Spanish, with vowels on the right hand side instead of directly below. This is not significant, but an artifact of the process. In contrast, the earlier algorithm showed a slight rotation between English and German, and that could be significant. The interpretation of the diagrams produced by factor analysis is much harder than the earlier diagrams, but they do demonstrate the real distance between letters that appeared close together in the earlier diagrams. Thus in Spanish we can see that 's' and 'z' are not adjacent to 'p' and 'b' but closer to 'h'. In the earlier English diagram 'g' and 'h' appeared as neighbours, while here they are much more distant. A comparison of the two diagrams for English and the two diagrams for Spanish reveals more about the nature of each language, but also shows how poorly any two dimensional picture captures the multidimensional reality.

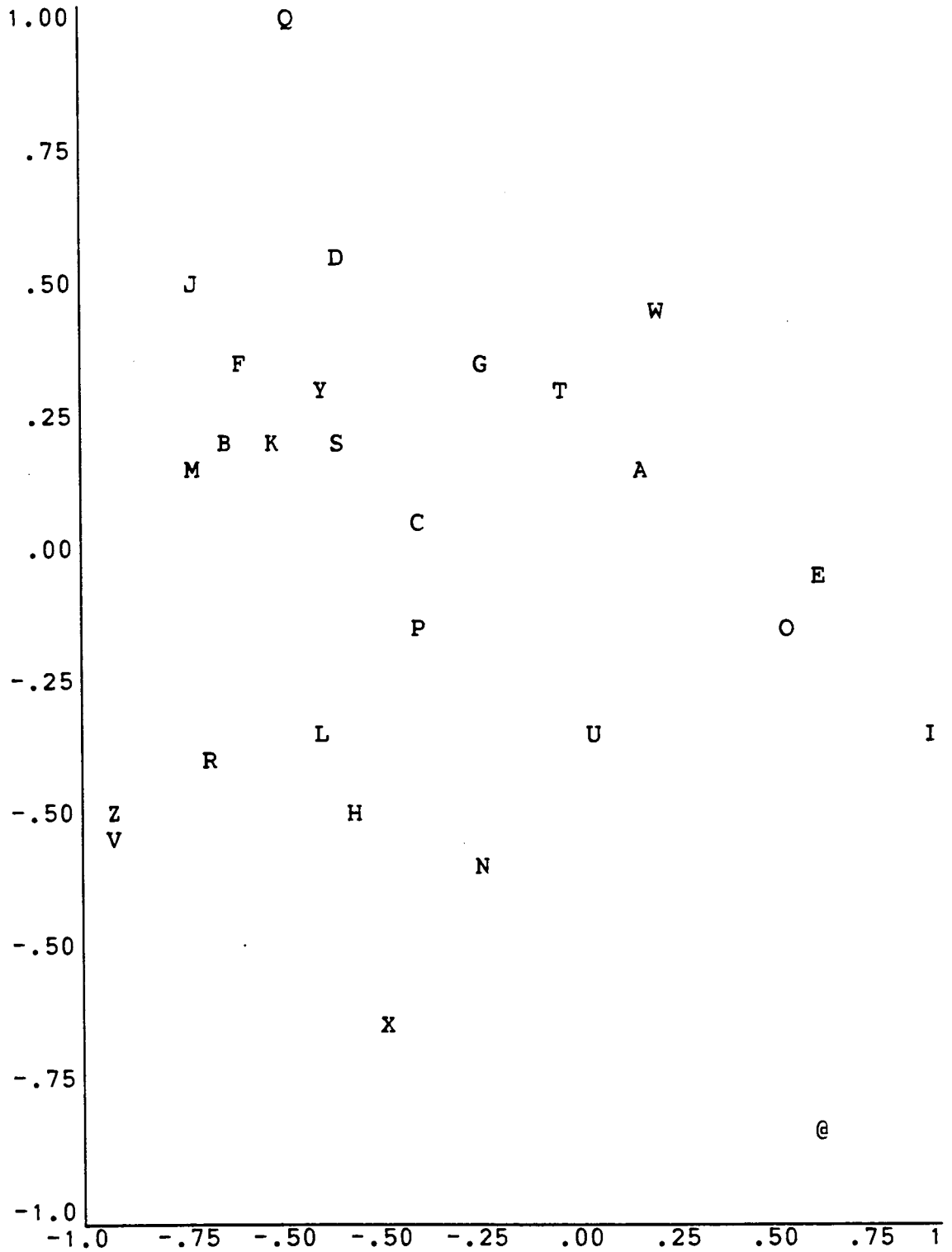


Figure 23. Analysis of English Text Sample

3.2 General Conclusions

I have shown that information about the sounds of alphabetical symbols may be obtained from a study of printed text, and suggested that this may eventually be extended to a theory that all information necessary to understand and use a human language may be easily extracted from a long enough sample of the language.

The existence of methods such as described here has many important consequences. The possibility of performing a complete analysis of a sample of human language without the aid of additional information provides an interesting model of human language acquisition. It also shows how the linguist may describe the grammar of a language without learning the language first.

Bloomfieldian linguistics gave the linguist a well-defined and limited input: actual utterances of native speakers. From these utterances, observed in context, the linguist was supposed to be able to develop a grammar of the language. Any theory capable of showing how and why this could be done would also serve to explain language acquisition. The modern generative grammarian never needs to show how and why he can produce a grammar of a language, since he must in some sense "know" the grammar of the language before he starts. His work is only an externalization of what he has internally.

If human language processes are linear, then the investigation of language can proceed without any need for introspective judgements or reference to mental states.

The claim that human language use is a linear process is based on the narrower claim that spontaneous unrehearsed (real-time) speech is a linear process, because in such circumstances the constraints of time and social pressure leave little alternative but immediate acceptance of the promptings of memory. This leads to the notion that a stream of linguistic items in real-time speech is somewhat self-determining, or tends to run its course, with only limited direction from the will.

It was noted earlier that linear devices produce outputs which is related to the input in a certain limited way. In dealing with human language use and treating it as a linear operation, I have suggested an inevitable resemblence between inputs and outputs. An ordinary person exposed to a large number of different inputs will tend to use synonymous expressions since his various inputs provide him with various different ways of saying the same thing. Ordinarily the choice of one synonym rather than another will depend on context.

The generative model of language, which focuses on the activities of a message source, differs from the earlier, behaviourist model in which a person is treated as a "black box" which responds in observable ways to observable stimuli. Although I am suggesting that we mimic others, and emphasize the close correspondance between stimuli and response in language, I do not agree with behaviourists who suppose that stimuli somehow determine response, or that responses may be controlled by conditioning or other manipulation of stimuli. The closest traditional theory of psychology to the views presented here is

not behaviourism, but rather associationism, particularly the simple early associationism of James Mill.

The theory presented here can most easily be described as a theory of associative memory and its consequences. Associative memory relates to another aspect of memory, data compression. Data compression involves recognizing the patterns that occur in data because of redundancy, and re-organizing the data so that these redundancies can be eliminated. The reconstruction of data from compressed storage involves the recreation of expected patterns of redundancy. Imperfect reconstruction of data leads to an associative memory where seeing or hearing one of pair of items may bring to mind the other. A consequence of this is a tendency to reproduce familiar patterns of co-occurence.

In real-time speech utterances must be produced without reflection. There is little option but to utter whatever associative memory produces. Note, however, that one's own utterances are also part of the real-time context: the utterance of a single word itself changes the overall context and should be considered as a alteration in that context. The word one has uttered, together with the other contextual material stimulates the memory, and the need for quick continuity makes one accept the newly remembered material as one did the old.

A consequence of data-compression and associative memory for language is thus a tendency for certain linguistic items to occur together, which may broadly be called syntax.

The successful use of language begins with the ability to use words in context, and proceeds with the learning of what they mean. Rather than choosing to use expressions that most accurately reflect the meanings to be expressed, people seem often to use expressions they have heard, most often in contexts like those in which they heard the expressions used by other people. This may be seen as a result of the individual's associative memory. A person tends to associate certain utternaces with the contexts in which he has heard them. If the association between utterance and context is strong enough, similar contexts may evoke similar utterances on his part, even if the full semantic content of these utterances is unknown to him. Indeed, I would suggest that we rarely have any idea of the full content of what we say. In using language without knowing the exact meanings of what they say, people pass on information over and above their intended meanings. It is my contention that this flow of information is vitally important to the functioning of society as a whole.

It is common in sociology to consider human society as a vast network of interacting human beings. This societal information network is constrained by the same basic limitation as all networks: individual nodes need to communicate with many other nodes, yet they can interact directly with only a few of them. Rather than have each individual human being do as much information processing as possible, society functions by having each person spend much of his time just acting as a channel for communications.

Information flows through people when they use words and expressions, accents and intonations, without necessarily understanding much of their semantic and social implications. Each person passes on information about the people he has listened to. A person may be able to consciously control the extent to which he mimics others, but this is only a process of regulating a flow of information between other people. Regardless of how a person may attempt to control his speech, he still acts as a communication channel between the people he listens to and other people he later talks to.

If it was important to society that people controlled the statistical properties of their speech, such abilities would probably have evolved at the time man's ancestors became social animals and began to develop speech. On the other hand, the human inability to control statistical properties of speech may lead to a flow of information that would not otherwise occur. A society may need a flow of information between people who cannot directly communicate with one another, or who would not choose to do so. Existing limitations on human language processing may serve to guarantee a flow of information between individuals regardless of their intentions.

Chomsky (1975:40) has described language as a uniquely human facility, without precedent in the animal world. He has not been able to give any satisfactory account of how such an ability could have evolved. I suggest instead that we use language in imitation of other people to whom we have listened,

and that this is just an elaborate development of the familiar processes of mimicry and imitation by which advanced animals coordinate their behaviour with the observed behaviour of others.

For mimicry to be valuable, it has to occur at the right time and place, and so the evolution of behaviour must have involved an increasing awareness of environment and social circumstance. What differentiates the human language facility from the simple mimicry of the parrot is not, therefore, the possession of some complicated system of grammatical rules, but merely an extreme sensitivity to context, so that utterances appropriate to each particular social situation are more likely to be reproduced.

3.3 Summary of Argument

This thesis includes several arguments leading to conclusions that should be listed together:

1. The notion of linearity has been restricted because of its association with systems of equations.
2. The term linear means pseudo-invertable
3. The theory of perfect coding and decoding is the theory of invertable devices, which is just the theory of groups.
4. A theory of language must be a theory of imperfect coding and decoding devices, i.e. pseudo-invertable devices, i.e. linear devices.
5. Linear devices can code and decode by convolution and deconvolution.
6. The grammar of a device is its system transfer function.
7. One way of finding system transfer function is with simulation. We can try to produce a device which has the same output as the system being studied. For this purpose the input is usually assumed to be uncorrelated.
8. Markov devices are implemented by applying an uncorrelated input to a filter, and thus are simulations of human language processes that should have similar system transfer functions.
9. Chomsky's objections to finite-state Markov sources can be overcome by treating these devices as filters.

10. Chomsky's classification of grammatical devices is irrelevent to linguistics. Instead the distinction between linear and nonlinear devices should be considered.
11. Finite-state machines may be implemented as linear devices.
12. Finite-state machines may be adequate as models of language if punctuation is added. Human languages do express punctuation, both prosodically and with punctuation words.
13. With punctuation language is associative and closed, and thus may be easily treated as an algebraic structure.
14. Finite state machines can be extended upward to Chomsky's other classes by adding feedback, or by rearranging feedback and output so that what is fedback is not just a copy of the output.
15. If this is done, hidden states and unstable states are added, thus better modelling human language.
16. These abstract machines can be thought of as linear corrective filters.
17. A linear corrective filter can function as an associative memory by 'correcting' current observations with a reminder of what is missing from them, i.e., what similar situations contained in the past.
18. Thus associationist psychology can be related to this theory of language.
19. A problem with associationism is the apparent need for

variable weights. How are these to be varied without non-linearity?

20. Instead of multiplicative weights, additive functions of past inputs can be used.

21. Output processes involve a matching, such as finding the best match of vector. This is a linear process. Addition of functions of past inputs can increase the probability of a vector being chosen as best match, and thus simulate multiplicative weighting.

22. Thus, we learn language by accumulating instances of possible outputs, which are stored associatively by context, with both storage and retrieval being linear processes.

BIBLIOGRAPHY

Albus, J.S. (1972). The Cerebellum: A Substrate For List-Processing In The Brain. In H. W. Robinson & D. E. Knight (Eds.), Cybernetics, Artificial Intelligence and Ecology. New York: Spartan Books.

Albus, J. S. (1975). A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)". Jounal of Dynamical Systems, Measurement and Control, 97(3), 220-227.

Albus, J. S. (1981). Brains, Behaviour, and Robotics. Peterborough, NH: Byte.

Anderson, J. R. & Bower, G. H. (1973). Human Associative Memory. Washington, DC: Winston and Sons.

Benwick, L., le Fay, M., & Knight, G. (1976). Camelot 1968. In J. D. McCawley (Ed.), Notes from the Linguistic Underground. New York: Academic Press.

Berlinski, D. (1976). On Systems Analysis. Cambridge, MA: MIT Press.

Beth, E. W. (1964). The Foundations of Mathematics (rev. ed.). Amsterdam: North Holland.

Brigham, E. O. (1974). The Fast Fourier Transform. Englewood Cliffs, NJ: Prentice-Hall.

Bruns, G L. (1974). Modern Poetry and the Idea of Language. New Haven, CT: Yale University Press.

Carnap, R. (1958). An Introduction to Symbolic Logic. New York: Dover.

Chapeney, D. C. (1973). Fourier Transforms and their Physical Applications. London: Academic Press.

Chomsky, N. A. (1955/1975). The Logical Structure of Linguistic Theory. New York: Plenum Press. (Unpublished manuscript circulated privately 1955).

Chomsky, N. A. (1957). (1957)Syntactic Structures. The Hague: Mouton.

Chomsky, N. A. (1961). On the Notion, "Rule of Grammar". In J. A. Fodor & J. J. Katz (Eds.), The Structure of Language. Englewood Cliffs, NJ: Prentice-Hall.

Chomsky, N. A. (1963). Formal Properties of Grammars. In R. D. Luce, R. R. Bush, & E. Galanter, Handbook of Mathematical

Psychology (Vol. 2). New York: John Wiley and Sons.

Chomsky, N. A. (1965). Aspects of the Theory of Syntax. Cambridge, MA: MIT Press.

Chomsky, N. A. (1975). (1975) Reflections on Language. New York: Pantheon.

Chomsky, N. A. & Halle, M. (1968). The Sound Pattern of English. New York: Harper and Row.

Chomsky, N. A., & Miller, G. A. (1963). Introduction to the Formal Analysis of Natural Languages. In R. D. Luce, R. R. Bush, & E. Galanter, Handbook of Mathematical Psychology (Vol. 2). New York: John Wiley and Sons.

Cohn, P. M. (1965). Universal Algebra. Dordrecht, Holland: D. Reidel Publishing.

Davis, P. W. (1973). Modern Theories of Language. Englewood Cliffs, NJ: Prentice-Hall.

Dineen, F. P. (1973). An Introduction to General Linguistics. New York; Holt Rinehart and Winston.

Duda, R. O. and Hart, P. E. (1973). Pattern Recognition and Scene Analysis. New York: John Wiley and Sons.

Ellson, D. G. (1949). The Application of Operational Analysis to Human Motor Behavior. Psychological Review, 61, 9-17.

Ellson, D. G. (1959). Linear Frequency Theory as Behavior Theory. In S. Koch, Psychology: A Study of a Science (Vol.2). New York: McGraw-Hill.

Faddeev, D. K. & Faddeeva, V. N. (1963). Computational Methods of Linear Algebra. San Francisco: W. H. Freeman.

Firth, J. R. (1935). The Use and Distribution of Certain English Sounds. In J. R. Firth, Papers in Linguistics 1934 - 1951. Oxford: Oxford University Press.

Firth, J.R. (1951a). Modes of Meaning. In J. R. Firth, Papers in Linguistics 1934 - 1951. Oxford: Oxford University Press.

Firth, J.R. (1951b). General Linguistics and Descriptive Grammar. In J. R. Firth, Papers in Linguistics 1934 - 1951. Oxford: Oxford University Press.

Foley. (1977). Foundations Of Theoretical Phonology. Cambridge: Cambridge University Press.

Gabor, D. (1949). Microscopy by Reconstructed Wave-fronts.

Proceedings of the Royal Socety, A197

Gill, A. (1962). Introduction To The Theory Of Finite-State Machines. New York: McGraw-Hill Book Company.

Graupe, D. (1972). Identification Of Systems. New York: Van Nostrand Reinhold.

Harman, G. (1972). Deep Structure as Logical Form. In D. Davidson & G. Harman (Eds.). Semantics of Natural Language. Dordrecht, Holland: D. Reidel Publishing.

Herdan, Gustav (1966). The Advanced Theory of Language as Choice and Chance. New York: Springer-Verlag.

Hockett, C. F. (1955). Manual of Phonology. Bloomington: Indiana University Press.

Hofstadter, D. R. (1979). Godel, escher Godel, Escher, Bach: an Eternal Golden Braid. New York: Basic Books.

Humboldt, W. von (1836/1971). Linguistic Variability and Intellectual Development (G. C. Buck and F. A. Raven, Trans.). Miami, FL: University of Miami Press.

Kain, R. Y. (1972). Automata Theory: Machines and Languages. New York: McGraw-Hill.

Katz, J. J. and Postal, P. M. (1964). An Integraded Theory of Linguistic Descriptions. Cambridge, MA: MIT Press.

Kneale, W. and M. (1962). The Development of Logic. Oxford: Clarendon Press.

Kohonen, T. (1977). Associative Memory, A System Theoretic Approach. New York: Springer-Verlag.

Kuhn, T. S. (1970). The Structure of Scientific Revolutions (2nd ed.). Chicago: University of Chicago Press.

Lakoff, G. (1972). Linguistics and Natural Logic. In D. Davidson & G. Harman (Eds.). Semantics of Natural Language. Dordrecht, Holland: D. Reidel Publishing.

Langer, S. K. (1957). Philosophy in a New Key (3rd. ed.). Cambridge, MA: Harvard University Press.

Langer, S. K. (1967). Mind: An Essay on Human Feeling. Baltimore, MD: Johns Hopkins.

Lindsay, P. H. & Norman, D. A. (1972). Human Information Processing. New York: Academic Press.

Lyons, J. (1966). Firth's Theory of Meaning. In C. E. Bazell (Ed.), In Memory of J. R. Firth. London: Longmans.

Lockwood, W. B. (1969). Indo-European Philology. London: Hutchinson.

McCawley, J. D. (1972). A Program for Logic. In D. Davidson & G. Harman (Eds.). Semantics of Natural Language. Dordrecht, Holland: D. Reidel Publishing.

Massimo Piattelli-Palmarini (Ed.) (1972). Language and Learning. Cambridge, MA: Harvard University Press.

McMahon, L. E., Cherry, L. L. & Morris, R. (1978). Statistical Text Processing. The Bell System Technical Journal, 57(6).

Miller, G. A. and Chomsky N. A. (1963). Finitary Models of Language Users. In R. D. Luce, R. R. Bush, & E. Galanter, Handbook of Mathematical Psychology (Vol. 2). New York: John Wiley and Sons.

Miller, R. L. (1968). The Linguistic Relativity Principle and Humboldtian Ethnolinguistics. The Hague: Mouton.

Naylor, A. W. & Sell, G. R. (1971). Linear Operator Theory. New York: Holt, Rinehart and Winston.

O'Connor, J.D. & Trim, J. L. M. (1953). Vowel, Consonant, and Syllable - A Phonological Definition. Word 9(2), 103-122.

Parkinson, G. H. R. (1966). Introduction. In G. H. R. Parkinson (Ed. and Trans.), Leibniz: Logical Papers. Oxford: Clarendon Press.

Postal, P. M. (1964). Limitations of Phrase Structure Grammars. In J. A. Fodor & J. J. Katz (Eds.), The Structure of Language. Englewood Cliffs, NJ: Prentice-Hall.

Pribram, K. H. (1971). Languages of the Brain. Englewood Cliffs, NJ: Prentice-Hall.

Quine, W. V. O. (1964). The Problem of Meaning in Linguistics. In J. A. Fodor & J. J. Katz (Eds.), The Structure of Language. Englewood Cliffs, NJ: Prentice-Hall.

Quine, W. V. O. (1966). The Ways of Paradox and other Essays. New York: Random House.

Quine, W. V. O. (1960). Word and Object. Cambridge, MA: MIT Press.

Redei, L. (1967). Algebra (Vol. 1). Oxford: Pergamon Press.

Roberts, E.W. (1972). Consonant and Vowel: A Re-examination. Lingua, 30, 141-202.

Sage, A. P., & Melsa, J. L. (1971). System Identification. New York: Academic Press.

Sanders, G. A. (1972). Equational Grammar. The Hague: Mouton.

Simmons, R. F. (1973). Semantic Networks: Their Computation and Use for Understanding English Sentences. In R.C. Schank & K. M. Colby (Eds.) Computer Models Of Thought and Language. San Francisco: W. H. Freeman & Company.

Steiner, G. (1975). After Babel. Oxford: Oxford University Press.

Suppes, P. (1972) Probabilistic Grammars for Natural Languages. In D. Davidson & G. Harman (Eds.). Semantics of Natural Language. Dordrecht, Holland: D. Reidel Publishing.

Vetter, H. J. (1969). Language Behavior and Communication. Itasca, IL: Peacock.

Wilks, Y. (1973). An Artificial Intelligence Approach to Machine Translation. In R.C. Schank & K. M. Colby (Eds.) Computer Models Of Thought and Language. San Francisco: W. H. Freeman & Company.