7

Morphological analogy: Only a beginning

John Goldsmith

All reasoning is search and casting about, and requires pains and application.
John Locke, *An Essay Concerning Human Understanding* (1975 [1690])

7.1 Introduction

The perspective that I will describe in this paper is the result of some work over the last ten years or so aimed at building an automatic morphological analyzer—that is, an explicit algorithm that takes natural language text as its input, and produces the morphological structure of the text as its output.[1] The main conclusion, as far as analogy is concerned, is that formal notions that correspond very naturally to the traditional notion of analogy are useful and important as part of a boot-strapping heuristic for the discovery of morphological structure, but it is necessary to develop a refined quantitative model in order to find the kind of articulated linguistic structures that are to be found in natural languages.

I take the perspective that the three principal tasks (we could call them the *first* three tasks) of someone who wishes to develop a theory of morphology that applies to natural languages is to develop an account for (1) the segmentation of words into morphs; (2) the description of a grammar to generate words, on the basis of the morphs, among other things; and (3) the labeling of morphs, in two different ways: (a) a labeling that indicates which morphs are different realizations of the same morpheme, and (b) a labeling that indicates the morphosyntactic feature representation of each morpheme. Of these three, I will focus on the first two, and of the first two, I will emphasize the first. I underscore this because if we were historians of linguistics in the future

[1] I am grateful to Juliette and Jim Blevins, to Susan Rizzo, and to anonymous referees for comments on the original version of this chapter.

looking back at what questions were the focus of discussion in the first decade of the twenty-first century, it would appear that the first question must have been settled, in view of how little discussion there is of it.[2] I mean very simply, how do we justify the statement (for example) that *books* is composed of two morphs, *book* and *s*, while *tax* is not? One of the reasons that the problem of segmentation is interesting is that we cannot call upon the resources within generative grammar that most of us are familiar with, and have grown dependent upon—which is to say, appeal to substance in an innate Universal Grammar. There is no plausible account of how speakers of English learn that "*ing*" is a suffix, while speakers of Swahili learn that "*an*" is a suffix, that appeals to a small list of discrete parameters, each with a small number of settings.[3] In fact, from a certain point of view, this is one of the reasons why the study of morphology so interesting: there is so much that must be learned.

I will begin with a discussion of the computational problem of *word* segmentation—that is, the problem of dividing a long string of symbols into words, with no prior knowledge of the words of the language. This is one of the problems that any child language learner faces. We will see that a large part of the difficulty that we run into when we tackle this problem derives from the importance of having a good model of morphology, without which all of our efforts to learn words would be in severe trouble. Rather than trying to solve both problems at the same time (the problem of word segmentation, and the problem of morphology induction), we will turn

[2] There is a perspective on word structure, articulated notably by Rajendra Singh and Sylvain Neuvel (Neuvel and Singh (2001), Neuvel and Fulop (2002)), which denies the existence of morphs and the internal segmentation of words. While I appreciate the force of their arguments, it seems to me that the same arguments against the decomposition of words into morphs holds, with essentially the same degree of conviction, against dividing sentences up into words—there are unclear cases, there is semantic noncompositionality in quite a few cases, and so on. But at the same time, it seems to me that linguists have to agree that concatenation is the preferred formal operation in both morphology and syntax, and the focus on segmentation into words and morphs can be understood as no more and no less than a consequence of that preference.

[3] There is a tradition of no great antiquity in linguistic theory of seeing the adult grammar as a collection of objects selected from a fixed, universal inventory of objects, rather than as an algebraic representation of some sort whose length is in principle unbounded. The first explicit mention of this, as far as I know, is found in David Stampe's work on natural phonology in the early 1970s (see Stampe (1980) [1972]), followed by Daniel Dinnesen's atomic phonology (see Dinnsen (1979)); the strategy was adopted in Chomsky's principles and parameters at the end of the 1970s, and it has never left the charts since then. It gained renewed vigor with the rise of optimality theory in the 1990s. Its appeal is no doubt due to the pious hope it has been known to inspire that the problem of language learning may turn out to be trivial, because the differences between languages will amount to a small number of bits of information. I find this sad, in part because, if we can't count on linguists to tell the world about the richness and variety found across humanity's languages, there is no one else to do it. It's doubly sad, in that even if it *were* the case that learning a language could be modeled as being much like selecting a set of, say, 50 items out of a universal set of 1,000, we would still need to do some heavy lifting to produce an account of learning; since there are some $\frac{1000!}{50!\ 950!}$ ways to do that, the fact that this is a finite number is not much consolation. I will return to this in the conclusion.

specifically to the task of discovering the morphology of a language with no prior knowledge of the morphology, but with prior knowledge of where word boundaries are (as if we had already solved the word segmentation problem), and discuss the role that analogy plays in this latter task. Naturally, we would like to merge these two tasks, and present an algorithm that takes an unsegmented segment stream as input and produces both a word list and a morphology; we are not yet able to accomplish that (though I suspect we have the tools at our disposal now to tackle that problem). I would like to emphasize, however, that the materials on which we base our experiments are not prepared corpora or toy data; they are in every case natural materials from natural languages.

There is a more general point behind my account as well, which deserves at the very least a brief presentation before we settle into a discussion of a specific problem. It is this: the present paper assumes that we can specify a scientific goal for linguistics which is independent of psychology, and which depends only on computational considerations. Being independent of psychology, it does *not* presume to tell psychologists what conclusions they will or should reach in their exploration of the human mind and brain, nor does it depend on those explorations. Its premise is very simple: given a particular corpus from a language (that is, a finite sample, which can be as little as a few thousand words, or as large as the internet as of some moment in time, like today), the goal is to find the best grammar (or set of grammars) that accounts for that data. This suggestion is only as useful as our ability to explicate what it means for a grammar *G* to account for a set of data, or corpus, *C*, and we will define this as the probability of the grammar *G*, given the data *C*; and we will see below that by this we shall have meant the grammar *G* such that its probability (based on its form) multiplied by the probability that *G* assigns to *C*, is the greatest. How such a view is possible and reasonable will become clearer shortly.[4]

Before proceeding any further, I would like to say what I mean by analogy in morphology. Unless specified otherwise, I will assume that our goal is to analyze the internal structure of words, and also that we actually know where words begin and end in the sound (or letter) stream of the language we happen to be looking at. In fact, I will assume that our problem is to find internal structure when presented with a word list in a language. In traditional terms, *book* : *books* :: *dog* : *dogs* would constitute an analogy; so would *jump* : *jumped* : *jumping* :: *walk* : *walked* : *walking*. A more perspicuous way to look at this sort of analogy is as in (1), which we call a "signature"; a computer

[4] This notion is also presented, in greater detail, in Goldsmith (2007).

scientist would prefer to represent the same data as in (2), which he would call a representation of a finite state automaton (FSA).

$$\left\{\begin{matrix} \mathit{walk} \\ \mathit{jump} \end{matrix}\right\}\left\{\begin{matrix} \emptyset \\ \mathit{ed} \\ \mathit{ing} \end{matrix}\right\} \tag{1}$$

$$\begin{tikzcd}
\bullet \arrow[r, bend left, "walk"] \arrow[r, bend right, "jump"'] & \bullet \arrow[r, bend left, "\emptyset"] \arrow[r, "ed"] \arrow[r, bend right, "ing"'] & \bullet
\end{tikzcd} \tag{2}$$

But before we talk about morphological analysis, let us turn first to the problem of word segmentation.

7.2 The problem of word segmentation

In the mid-1990s, Michael Brent and Carl de Marcken (both graduate students in computer science at the time working with Robert Berwick at MIT) developed computational methods for inferring word boundaries in a continuous stream of discrete symbols, relying on Minimum Description Length (or *MDL*) analysis (Brent (1999), de Marcken (1996), Rissanen (1989)). Their projects could be interpreted (as they did interpret them) as representing an idealization of how a child can learn the words of a language when exposed only to a stream of phonemes. This is the *word segmentation problem*: how to find words in a larger stream of symbols. Now, there are two fundamentally different approaches that one could take in dealing with the word segmentation problem (and one could certainly adopt both approaches, since they are not incompatible): one can either focus on finding the *boundaries* between words, or focus on finding words *themselves* in the stream, the sequences of recurring symbol strings, and inferring the boundaries from knowledge of the words. I think that there is a widespread (and natural) tendency to feel that the first of these two methods (finding cues in the signal that show where the boundaries between words are) is the more appealing way to approach the problem, perhaps on the grounds that you cannot take the second approach without engaging in some kind of inappropriate circular reasoning. This intuition is probably encouraged, as well, by the observation that in a good number of European languages, there are relatively straightforward superficial phonological cues to mark the delimitations between words, such as can be found in words in which the initial syllable is regularly stressed (as in Finnish, and as was once the case in German), or in which the penultimate syllable is stressed.[5]

[5] As an aside, I would mention my belief that this approach is hopeless as a general solution to the problem of word segmentation. The reason for this pessimistic view is that the difference in

The second approach, as I noted, is to say that we will *first* find the words in the signal, and *then* divide the signal up into words in the most likely way based on that knowledge of the words, along with the assumption that the speech signal can be partitioned without overlap into a succession of words. But how can this kind of learning be done?

I will give a brief summary here of the Brent-de Marcken approach to answering this question, based on MDL modeling. My account leans more heavily on de Marcken's specific approach than on Brent's, but it is a simplification of both, and the reader who would like to learn more is strongly advised to read the original works.

Minimum description length modeling was first developed by the Finnish-American statistician, Jorma Rissanen, notably in a book published in 1989 (Rissanen (1989)). The question he is concerned with is not specifically linguistic at all. It is simply this: given a body of data, how can we be sure to extract all and only the regularities that inhere in the data? We want to fit the model to the data, or the data to a model, and we want neither to overfit nor to underfit. Underfitting would mean failing to extract some significant regularity in the data; overfitting would mean misinterpreting something that was, in some sense, accidentally true of the data which was sampled, but would not be true of a larger sample from the same source.

Rissanen's approach is inherently probabilistic in two ways. To explain what these ways are, I shall discuss the problem of word segmentation in particular, even though Rissanen's approach is very general and was not developed with linguistic problems in mind. The first way in which the MDL approach is probabilistic is that an MDL analysis is a model (or grammar) that assigns a probability to every conceivable string of phonemes (or letters, if we are working with a language sample from a written source). This is a stringent condition: a probabilistic model is by definition one which assigns a non-negative number to every possible input, in such a fashion that the grand total of the probabilities adds up to 1.0—and this must be true even if the set of possible inputs is infinite (which is virtually always the case). Probability is thus *not* a measure of something like uncertainty or randomness; if anything, imposing the condition that the model be probabilistic imposes a very tight

probabilities that such approaches can assign to cuts in different places in a sound stream are far too small to allow a successful overall division of the stream to be accomplished in a local way, that is, based only on local information. The problem can only be solved by maximizing the probability of a parse over the longer string, which allows us to take into account the probabilities of the hypothesized words, as well as the conditional probabilities of the hypothesized words. To put this in a slightly different way, in order to segment a stream into words, it is not sufficient to have a model that predicts the phonetics of the word boundaries; one must also have a language model, assigning a probability to the sequence of hypothesized words. The interested reader can find a survey of much of the material on segmentation in Goldsmith (2009). See also Roark and Sproat 2007.

overall constraint on the system as a whole. In the language of probability, we are required to specify ahead of time a sample space and a distribution over that sample space; the distribution is essentially a function that maps a member of the sample space (or a subset of the members of the sample space) to a real number, in such a way that the whole sample space maps to 1.0.

The second way in which an MDL analysis is probabilistic is more abstract. We set a condition that the grammars *themselves* are the subject of a probability distribution; which is to say, every possible grammar is assigned a probability (a non-negative real number), subject to the condition that these probabilities sum to 1.0—and this must be true even if the set of possible grammars is infinite (which is virtually always the case). The reader may note that this condition puts MDL within the broader context of approaches which includes Bayesian approaches to modeling; MDL puts the priority on the quantitative notion of encoding, both regarding the data and the grammar, but there is an overall commonality from a distant enough perspective.

Although it may not sound like it at first, this second condition is very similar in spirit to Chomsky's view of grammar selection in early generative grammar (that is, in classical generative grammar (Chomsky (1975 [1955]), which in the late 1970s many generative grammarians abandoned—after little discussion—in favor of the principles and parameters approach (Chomsky and Lasniik (1977))). According to this perspective, the primary goal of linguistic theory is to make explicit a formalism for grammar writing, but not just *any* formalism. The goal was a formalism with which predictions (or, more modestly, claims) could be made as to which grammar was correct among a set of grammars all consistent with the given data; those predictions would be based purely on the length of the grammar in the some-day-to-be-discovered formalism.

MDL employs a few simple ideas to assign a probability to a (potentially infinite) set of grammars, and we should at least sketch these ideas. Perhaps the most important is what is known as Kraft's inequality. Kraft's inequality holds for uniquely decodable codes, but we will consider (as does most MDL modeling) a special case of that—those codes which are said to respect the prefix condition. The term *coding* here should simply be interpreted as meaning something like *formalized as a grammar*, and in general we want to consider the class of all grammars that are permitted by a certain formalism. The prefix condition sounds innocuous: it says that there are no two grammars (*G* and *H*, say) which have the property that *H* equals all of *G* plus some additional material (as computer scientists put it: there are no two grammars *G* and *H* such that *G* is a prefix of *H*—but remember that computer scientists just use "prefix" to mean a substring that starts at the beginning of some other string). Another way to put it is this: when you are reading a grammar, you know when you reach the end of it. (The condition seems innocuous, but its consequences are major, for reasons that we will not go into here.)

Kraft's inequality says that if a set of strings (here, grammars) does indeed respect the prefix condition, then we can assign a probability to each string (grammar) S equal to $2^{-length(S)}$. Why the number 2 here? I have assumed (as computer scientists tend to) that we encode the grammar using strictly binary encodings, the way a computer does, using only 0's and 1's. If we want to use a vocabulary like the Latin alphabet, then the base is going to be 26—or more likely 27, if we include a punctuation symbol, like space,[6] and so below I will replace "2" by "27." If the length of a grammar is 100 0's and 1's, then we assign it a probability of $\frac{1}{2^{100}}$; if it's 100 *letters*, then we assign it a probability of $\frac{1}{27^{100}}$. Unless we're very careful with our assignment of lengths, this quantity (based solely on grammar length) will sum to a finite number less than 1 (call it k); and then, to turn these numbers into true probabilities, we divide each of them by k, so that the sum totals 1.0.

In short, with a very mild condition (the prefix condition) imposed, we can easily specify a natural probability distribution over the infinite class of grammars, according to which a shorter grammar is a more probable grammar. In fact, if grammar G has length g, and grammar H has length h, then the ratio of their probabilities is simply $2^{(g-h)}$ if binary encoding is used, and $27^{(g-h)}$ if the Latin alphabet is employed.

Now we take two further steps. The first involves Bayes' rule, which is nothing more than an algebraic restatement of the definition of conditional probability. The second involves the assumption that there is a single correct answer to our question.

Bayes' rule says that (in the case that we are considering) the probability of a grammar, given our corpus, is closely related to the probability of the grammar, given the corpus, as follows:

$$pr(G|D) = \frac{pr(D|G)pr(G)}{pr(D)} \tag{3}$$

The left-hand side refers to the probability of a grammar G, given the data D at hand (i.e., the corpus), while the right-hand side is the product of the probability of the corpus assigned by the grammar G, times the probability of the grammar, divided by the probability of the data. Since our goal is to find

[6] There is a fine line here between clarity of exposition and accuracy of modeling. In general, we *don't* want to use special boundary symbols to demarcate the ends of representations, because this is typically a wasteful and inefficient way of marking boundaries; an encoding which respects the prefix condition is better. But ease of exposition will sometimes trump formal niceties in this chapter.

the grammar whose probability is the greatest (given the data at hand, and what else do we have other than the data at hand?), we can interpret (3) to mean: find the grammar *G* for which this quantity is the greatest. The denominator, $pr(D)$, is perhaps the hardest to compute, but we do not in fact need to calculate it, because it is a constant. Since we have just finished discussing how to calculate the probability of the grammar *G*, based on its length, calculating $pr(G)$ is not a problem. And calculating $pr(D|G)$ is not a problem, either, since we have assumed from the start that our model is probabilistic, which is to say, that it assigns a probability to every conceivable corpus. So in the end, our task simply boils down to this: find the probabilistic grammar *G* such that the probability of the corpus, given the grammar, times the probability of the grammar itself, is the greatest.

Brent's and de Marcken's insight was that the method that we have just described could be applied to the problem of word segmentation and lexicon induction. We need to do three things: first, figure out how a lexicon (with its probability) actually assigns a probability to any corpus; second, figure out how to associate a lexicon with a length, so that we can in turn assign it a probability; and third, figure out how to actually come up with a candidate lexicon, along with probabilities assigned to each word in the lexicon. It turns out that none of these is too difficult, at least as a first approximation.

First, how do we assign a probability to a corpus *D*, given a probabilistic lexicon? We need to take into consideration the fact that there will, generally speaking, be many ways of parsing a corpus up into words. If all we know about English is its words (and nothing about syntax, meaning, and so on), then a string like: THISMEANSTHAT that can be divided up in many ways. There is THIS-MEANS-THAT, but then (since every individual letter can be used as an individual word in languages, in general), there is also THIS-ME-AN-S-THAT, and T-HIS-ME-AN-S-T-HAT, and many others. So first of all, we make the assumption that only one parse of a given corpus is actually correct,[7] and that the parse that is assigned the highest probability by our corpus is the correct one. And the probability assigned to a given parse is defined as the product of two factors: the first is the probability that the corpus has exactly as many words in it as the parse has pieces, while the second is the product of the probabilities of all of the words in the parse. In

[7] That this assumption is a bit too strong is illustrated by the ambiguity of phrases like "cookmea-napplesauce", which has perhaps two reasonable parses: *cook me an apple sauce* and *cook mean apple sauce*. The reader is invited to construct similar examples in other languages.

the case of THIS-MEANS-THAT, the probability of that parse is equal to the probability that a string has three words in it, times the product of the probabilities of each of the three words *this*, *means*, and *that*.[8]

Second, what is a lexicon's length? If we define a lexicon as a concatenation of words, then as long as we separate each of the words by a space, the words satisfy the conditions for Kraft's inequality, and we can assign a (prior) probability to a lexicon equal to 1 divided by 27 raised to the power of the length of the lexicon, in letters: $\frac{1}{27^{length(lexicon)}}$.

Third, how do we *find* a lexicon, given a corpus? We proceed in a bottom-up fashion, assuming initially that the lexicon consists of all the letters of the corpus. Then we iteratively repeat the following process: we look at all "words" that appear next to each other in the corpus, and pick the most frequent such pair. (Initially, this may be T-H in the case of a written corpus of English, since our initial assumption is that the words of the lexicon are the letters of the language). We use our MDL criterion to decide whether to declare that T-H is really a word TH. Our MDL criterion is simply this: does the expression described in (3) increase when we add our candidate to the grammar? Does the probability of the corpus increase enough by the addition of TH (for example) to offset the decrease in probability of the lexicon that comes about from increasing its length (from 26 real members to 27, the alphabet plus TH)? If so, then we include the new member; if not, we leave the grammar as it is and try some different candidates. This process stops when there are no neighboring chunks in the corpus whose addition would increase the overall probability of the corpus.[9]

There is one more step that we need to take to appreciate the beauty of Rissanen's MDL framework. If we take the logarithm of both sides of equation (3) and multiply these two expressions by −1, we obtain the following quantity: $-log\, pr(D|G) - length(G) + log\, pr(D)$. The third term is a constant. However, the first term has a very real significance: it is called the *optimal compressed length of the data*, and the second term also has a real significance: it is, quite simply, the length of the grammar, which we use in order to evaluate how well the grammar succeeds at being a compact formulation. The first term, the optimal compressed length of the data, given the model, is a well-understood quantity expressing how well the model does at extracting generalizations from the data. Thus the task of finding the grammar that *minimizes* this quantity (*minimizes* instead of *maximizes* because we

[8] There are several ways to establish a reasonable distribution over number of words in sentence, but they do not bear on our discussion here.

[9] See the Appendix.

multiplied it by −1, and the logarithm function is monotonic increasing) is equivalent to finding the most probable grammar, given the data at hand.

We intend by this to mean what was suggested above: there are no constraints on the forms of possible grammars, above and beyond the condition that they be programs for a Turing machine, and thus are algorithms.[10] This means that the purpose of linguistic theory is to serve as a set of heuristics to help the linguistic scientist come up with a tight, snug grammar, given a set of data. MDL can determine which of a set of grammars is the best one, given the data; no feasible process can search all possible grammars, so there is no guarantee that another linguist will not come along tomorrow with a *better* grammar for the data. But it will be *truly* better, better as far as the length of its Turing machine program is concerned. We know that there is a best analysis (up to the unlikely possibility that two or more grammars have (along with the data) an equal description length), because the minimum description length will be some positive number *less* than the description length provided by the (dumb) grammar consisting of exactly the corpus with no internal structure (along with some reasonable closure conditions).

7.3 Success with word discovery?

How well does this method work? Anyone who has worked with corpora knows that, to some extent, an answer to this question depends heavily on the corpus used for training and for testing. In the case at hand, there is no training corpus as such; the input to the algorithm is a long string that has no indication of word boundaries, and the output is a guess (or prediction) as to where the word boundaries are, or should be. In view of the fact that the system has no prior knowledge of the language, the results are in some respects very impressive, but at the same time, when we look at the results with the eyes of a linguist, we quickly see some linguisticky things that have gone awry.

In Figure 7.1 is the beginning of a passage from the first 100,000 words of the Brown corpus and Figure 7.2 is the beginning of a similar passage from a Portuguese document.

Three things jump out when we look at these results. First, there are many errors caused by the algorithm finding "pieces" that are too small, such as *produc-ed*: it seems as if the system is finding morphemes in this case, while in

[10] The point may be purely terminological, but I would argue that the position I am describing clearly falls under the definition of generative grammar, at least as it was considered in Chomsky (1975) [1955]; algorithmic complexity is the simplicity metric utilized.

The Fulton County Grand Ju ry s aid Friday an investi gation of At l anta 's recent prim ary e lection produc ed no e videnc e that any ir regul ar it i e s took place. Thejury further s aid in term - end present ment s thatthe City Ex ecutive Commit t e e, which had over - all charg e ofthe e lection , d e serv e s the pra is e and than k softhe City of At l anta forthe man ner in whichthe e lection was conduc ted.

FIGURE 7.1 The first sentences of the Brown Corpus

De muitosoutros re curso s da fl o r esta ,não apenas folh as, fl ores era íz esma s também de se mente se da cas ca de árvo res re ti ram produto s medi cin a i s comos quai s se habitu aram nas u a s o li dão e nos seu s s o nhos a en fre nta ra s do ença s que hoje coma chega da dos branco s começa ma trata r comos re médi os da indústri a u r ba na–e que muitas vezes não produz em e feito.

FIGURE 7.2 The first sentences of a Portuguese document

other cases it is finding words. Second, in some cases the algorithm finds pieces that are too big: they are "pieces" like *forthe* which occur together often enough in English that the algorithm erroneously decides that the language treats them as a word. Third, there are far too many single letter words: we need a prior probability for word length that makes the probability of one-letter words much lower.

We will focus here on just the first of these points. Why should the system find morphemes rather than words some of the time? The answer is perhaps obvious: the system that we are considering is nothing more than a lexicon, bereft of any ability to find structure in the data other than frequency of appearance of strings of various lengths. There is no ability built into the system to see relationships between words, nor any ability to see that words may enter into relationships with the words around them. We need to add linguistic structure to this approach, then. And that is what we turn to now.

7.4 The *Linguistica* project

I have been working since 1997, along with Colin Sprague, Yu Hu, and Aris Xanthos, on the development of a software package, *Linguistica*, whose primary goal is the automatic inference of morphological structure on the basis of an unmodified sample corpus from a real language, and whose method is MDL as we have described it in this chapter; see http://linguistica.uchicago.edu[11]

A big, and I would say controversial, assumption made by the *Linguistica* project is that meaning can be ignored in the process of inferring or inducing the morphological structure of a word or a language. The fact is, the procedures we have explored make little or no reference to meaning. Any successes that we achieve can be interpreted as showing that reference to meaning is not *necessary*, but we certainly cannot infer that human language learners do not use meaning in their search to discover language structure. It is natural to interpret our project as an effort to figure out, from a linguistic point of view, exactly *where* a learner, one who has access neither to a rich innate component nor to the meaning of utterances, will fail.

In some ways, the work that I am describing could be viewed as a neo-Harrisian program, in the sense that Zellig Harris believed, and argued, that the goal of linguistic theory was to develop an autonomous linguistic method of analyzing linguistic data, in which the overall complexity of the grammar was the character that the linguist would use in order to evaluate competing analyses, and in which the linguist was, in the final analysis, more interested in the methods of analysis than in the analysis of any particular language.[12] As long as we are clear what we mean by the term *discovery procedure*, it would be fair to say that this work aims at developing a discovery procedure for morphology. While it does not propose a simple step-by-step process for this end, it does propose something so close to an algorithm as to be indistinguishable from a computer program—which is why it has been relatively easy to encode the proposals as computer code which can be tested against small and large natural language corpora.

7.5 MDL, grammar simplicity, and analogy

One way to summarize what MDL methods have in common is to say that they seek to extract redundancy in the data. In the case of word segmentation, the redundancy is the reappearance of the same substrings on many occasions, while in the case of morpheme discovery, it is the reappearance of morphemes under quite particular and restricted conditions. What I will describe here is a considerable simplification of the model as it actually works, and the reader can find detailed discussion in Goldsmith (2001, 2006). As we saw above, the prior probability that is assigned to a grammar is based entirely on its

[11] See Goldsmith (2000, 2001, 2006).

[12] See Goldsmith (2005) for a recent discussion.

length, quite literally, and hence any redundancy in the formulation of a grammar leads to a heavy cost paid by the grammar, in terms of the lowering of the probability assigned to it. Conversely, a grammar which has been shortened by the elimination of redundancy is assigned a considerably higher probability. And, as we will see, analogy is one essential way in which redundancy can be discovered by the language learner.

The basic idea is this: when sets of words can be broken up into two pieces in precisely parallel ways (as in the signature shown in (1), repeated here as (4)), we can extract measurable redundancies. Here, we have taken the six words *jump*, *jumped*, *jumping*, *walk*, *walked*, and *walking*, and observed that there is a pattern consisting of two distinct stems, and three distinct suffixes, and all combinations of stem and suffix appear in our data set.

$$\left\{\begin{array}{c} \mathit{walk} \\ \mathit{jump} \end{array}\right\}\left\{\begin{array}{c} \emptyset \\ \mathit{ed} \\ \mathit{ing} \end{array}\right\} \tag{4}$$

Before any such analysis, we were responsible for encoding all the letters of the six words, which comes to forty letters (including a final space or word boundary), while after we extract the regularity, only sixteen letters need to be specified (again, counting a boundary symbol along with each suffix).

In somewhat more useful—that is, generalizable—terminology, we can describe this data with a finite state automaton (FSA), as in (2), repeated here as (5).

$$\begin{tikzcd} \bullet \arrow[r, bend left, "walk"] \arrow[r, bend right, "jump"'] & \bullet \arrow[r, bend left, "\emptyset"] \arrow[r, "ed"] \arrow[r, bend right, "ing"'] & \bullet \end{tikzcd} \tag{5}$$

To encode this, we need a formal method for describing the three states and their transitions, and then we need to label each transition edge; we have already seen a simple (and, as it turns out, overly simple) way of measuring the complexity of the labels, which was by counting the number of symbols. We will ignore the computation of the complexity of the FSA itself; it is very simple from a technical point of view.[13]

[13] Each FSA consists of a set of pointers to nodes, along with labels that are themselves pointers to strings. A maximum likelihood model provides probabilities in each of those two domains; the complexity of the overall FSA is the sum of the inverse log probabilities of all of the pointers in the representation.

This overall system can then naturally be regarded as a device capable of expressing morphological analogies of the *book* : *books* :: *dog* : *dogs* sort. How does it operate in practice? Does it work to find real linguistic morphological regularities?

The answer, in a nutshell, is this: we can find patterns, locally and in the small; but a very large proportion of them are spurious (that is to say, linguistically wrong and irrelevant) unless they participate in larger patterns of the language as a whole. An example of a linguistically real discovery is as in (4) or (5), and a spurious example is as in (6), which captures the nongeneralization inherent in the words *change, changed, charge, charged*, or (7), which captures the nongeneralization inherent in the words *class, cotton, glass, gotten* (and I could offer dozens of examples of this sort from any language of which we have a few thousand words in computer-readable form: it was not I, of course, who discovered these patterns, but rather an over-eager analogy-seeking computer program):

$$cha\left\{\begin{matrix} n \\ r \end{matrix}\right\}ge\left\{\begin{matrix} \emptyset \\ d \end{matrix}\right\} \tag{6}$$

$$\left\{\begin{matrix} c \\ g \end{matrix}\right\}\left\{\begin{matrix} lass \\ otten \end{matrix}\right\} \tag{7}$$

What is wrong with the spurious generalizations in (6) and (7) is that the proposed morphemes do not appear outside of this generalization, more generally in the language. Analogy, as we see it here, is an excellent and important source of hypotheses, but it is not more than that. We need to develop means (and, it appears, largely formal means) to evaluate the hypotheses suggested by analogies.

The use of Minimum Description Length analysis provides at least a part of the response to this need, and it sheds some interesting light on the role played by information theory in linguistic description. Embedded within the work cited above by de Marcken is the key insight formalized by the use of information-theoretic formalisms—namely, that reuse of a grammatical object (such as a morpheme, a context, or anything else) is the best kind of evidence we can have of the linguistic reality of the object. What makes the *n*, *r* pairing in (6) linguistically irrelevant is the small number of times it is found in the linguistic analysis of English—unlike the ∅, *d* pairing, but like the *c*, *g* pairing in (7).

But this should not lead us to thinking that we simply need to count occurrences and look for some magic threshold count, because information theory provides a much better method for understanding what is at play. The key point is this: the edges in the finite state automaton in (5) should be understood not as being labeled with strings of phonemes, but rather as being labeled by pointers to morphemes in a separate inventory of morpheme spell-outs. This simple formal decision has two consequences. The first is a consequence that comes from information theory: the complexity (in quantifiable bits) of a pointer to a morpheme is directly controlled by the frequency with which a morpheme is used throughout the grammar. The second is that we arrive at a natural understanding of the view, famously voiced by Meillet, that language is a system in which everything is interconnected.[14]

The decision to label edges of a morphology with pointers rather than phonic substance makes strong predictions: strong enough to build a program that figures out the structure by itself, without human oversight. *Linguistica* discovers affixes by seeking robust clusters of stems and affixes, such as the large set of stems in English that take exactly the suffixes ∅, *ed*, *ing*, *s*. But what of stems that occur with an idiosyncratic set of affixes, a set of affixes shared by no other stem? Consider the examples in (8) and (9).

$$act\left\{\begin{matrix}\emptyset\\ ed\\ s\\ ion\end{matrix}\right\} \tag{8}$$

$$car\left\{\begin{matrix}d\\ e\\ l\\ p\end{matrix}\right\} \tag{9}$$

Each of these signatures is an example of a stem that appears with exactly four suffixes in a pattern shared by no other stem in a particular corpus. But the information-theoretic cost of building a pattern with the suffixes in (8) is much less than that of building the pattern shown in (9)—not because of the number of letters (phonemes) in each case, but rather because /*l*/ and /*p*/ are both rare affixes in English (note: *affixes*, not phonemes). An affix that occurs

[14] In particular, "Comme pour tout autre langage, les différentes parties du système linguistique indo-européen forment un ensemble où tout se tient et dont il importe avant tout de bien comprendre le rigoureux enchaînement" (Meillet (1915) p. x).

on one word in a lexicon of 20,000 words will "cost" approximately *log*$_2$ 20,000 bits (about 14 bits), while a suffix that occurs on 1,000 words will cost about 4 bits—a very large difference, in the event; and the cost of positing /*l*/ and /*p*/ as affixes outweighs the gain saved by positing /*car*/ as a stem in (9). The same is not true of the case in (8), where the cost of building a subgeneralization to deal with the words based on the stem /*act*/ is much cheaper, because all of the observed suffixes are cheap, in an information-theoretic sense: they are independently used enough throughout the grammar that using them additionally in the creation of a new generalization costs the grammar very little. This implicit "thought process" is easy to formalize and to embed within an automatic morphological analyzer.

In Table 7.1, I have given some data from a sequence of steps of learning the morphology of the first 100,016 words of the Brown Corpus.

The first row in Table 7.1 shows the length of the "trivial" morphology at the beginning: it expresses the phonological cost (so to speak) of listing all 13,005 distinct words without any analysis: all words are stems, no stems are analyzed (we speak of "cost" to underscore the fact that we try to minimize this quantity). Row 2 shows the result of a relatively conservative effort to find signatures with several stems and several affixes, and we see that the information stored in the analyzed stems is now 53,835, while the information that we have taken away from the unanalyzed stems is greater: it is the difference between 486,295 and 390,160 (or 96,135). The additional infrastructure (affixes plus signatures) to accomplish this cost 1,220 + 22,793 (=24,013), for a total cost of 53,835 + 24,013 = 77,848. This cost (77,848) is much less than what was saved (96,135); the difference is 96,135 − 82,848 = 18,287. (Against this gain must be reckoned a slight decrease in the probability computed for the corpus.)

In the third, fourth, and fifth rows, we see the result of extending the discovery of signatures, stems, and affixes accomplished on the first pass to

Table 7.1 Description Length of morphology evolution during learning

Steps	Total	Unanalyzed stems	Analyzed stems	Affixes	Signatures
1. Before analysis	486,295	486,295	0	0	0
2. Bootstrap heuristic	468,008	390,160	53,835	1,220	22,793
3. Extend known stems and affixes	456,256	377,635	58,835	1,220	23,566
4. Find new signatures	434,179	320,405	74,440	1750	37,584
5. Find singleton signatures	429,225	235,390	128,830	1710	63,295

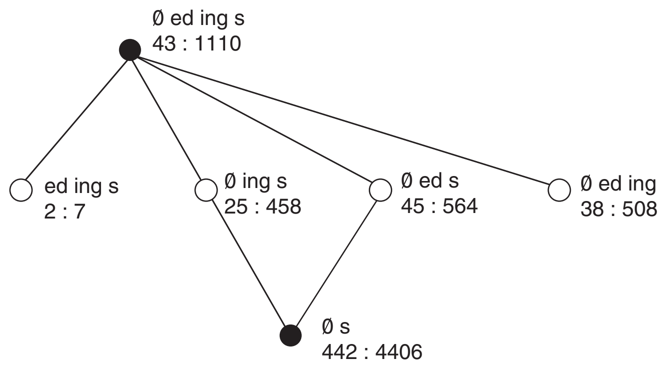
analyze words that were not initially analyzable. These are words for which the simple analogies of the first step were insufficient to uncover them, which include the discovery of patterns as in (8) and the rejection of those like in (9).

The algorithms explored in Goldsmith (2006) are remarkably good at discovering morphemes and morphological structure in a language with a complexity comparable to that of English. In the next sections, I will focus not so much on what they get right (which is better covered in the papers I have cited) but rather on where the challenges (some of them quite daunting) appear to be.[15]

7.6 The challenging of "collapsing" cases

Consider once more the case of English, where stems can be followed by a rather small set of affixes: verbs by {∅, *ed*, *ing*, *s*}, nouns by {∅, *s*}, adjectives by {∅, *er*, *est*}. In even a modest-sized corpus, we will find a large number of stems that appear with all of their suffixes inside the corpus. But in addition, we will find a good number of stems that only appear with a subset of their possible suffixes. In the simplest case, this is due to the fact that the stem did not appear very often in the corpus. This is illustrated in (10), where each node represents one of the signatures, or small FSAs, that we have considered, and it is labeled with its set of suffixes. Below the label are two numbers: the first indicates the number of distinct stems that occurred in the corpus with this set of suffixes, and the second indicates the total number of words that occurred with these stems and suffixes. The two filled nodes are the "saturated" ones in which, from a linguistic point of view, all the suffixes that could have appeared have appeared. The node on the top row has four suffixes; those on the middle row have three suffixes, each a subset of those of the node on the top row; and the node on the bottom row has two suffixes, a subset of the two nodes from which it hangs on the middle row.

[15] A reviewer of this chapter noted that "work on morphological processing (e.g. Baayen and Moscoso del Prado Martín (2005); Hay and Baayen (2005)) and [other work by Ernestus and Baayen]) suggests analogical relations are sensitive to semantic similarity, phonetic similarity, frequency effects, and more". The information-theoretic models of the sort discussed in the present chapter give a firm theoretical foundation for why frequency effects are found; the reason is that information links in a grammar contribute a measurable amount to the complexity of the system, and that amount is equal to the reciprocal of the logarithm of the element being linked to. In the morphological analyses that we have studied in the *Linguistica* project, phonetic similarity has never emerged as a factor which, if integrated, would allow for superior performance. The relevance of semantic information is a difficult question; while I believe that it is relevant and could potentially improve performance in many cases, it is not easy to integrate meaning into a learning algorithm in a way that does not beg the question of learnability by building in too much information and treating that information as if it had been observable.



(10)

We need a method that determines that the white nodes in (10) are only partial generalizations, while the filled nodes are complete. To be sure, I have expressed this in categorical terms, when it is clear (or it becomes clear, when we look at more data) that the distinction is a soft one, rather than a hard one—but discussion of this point would lead us afield. I will return below to this question in the context of a language like Swahili, where it becomes even more pressing. To rephrase the problem, we can ask, when we have two signatures that are partially identical and partially different, when is the similarity between them great enough to allow us to generalize the suffixes that are seen in one, but not in the other, to both of them? This remains an unsolved problem.

7.7 From analogy to algorithm

How does one actually *find* analogies along the lines of *book* : *books* :: *dog* : *dogs* in a language? It turns out that questions of this sort are not at all easy to answer, and a large part of the work devoted to the *Linguistica* project has been aimed at providing answers to this question. In this section, I will describe two problems that seem simple enough, and are certainly typical, and try to give a sense of why they are not as simple as one might expect them to be. The first example is the treatment of gender and plural marking of adjectives in French; the treatment of parallel forms in a number of other languages, such as Spanish, would be similar. The second is the treatment of morphological patterns in a rich system like that of the Swahili verb. "Treatment" in this context means the breaking up of the string into substrings corresponding to morphemes and the correct formulation of a finite-state automaton (or its equivalent) to generate the observed patterns. Thus we address both the first and the second question articulated in the first section of this paper.

As I noted above, some pre-generative linguists took such questions very seriously—notably, Zellig Harris (1955, 1967) did (but see Hafer and Weiss 1974). Harris apparently believed that he had solved the problem through the computation of what he called successor frequency (and predecessor frequency) in a large corpus. By successor frequency, Harris meant a characteristic of a specific string, in the context of a specific corpus: given a string S of length n (typically the first n letters of a word), one considers only the subset of words in the corpus that begin with the string S (computer scientists would say: consider the set of words with the prefix S—but then computer scientists use the term *prefix* rather differently than linguists), and then one asks: in this subset, how many different letters are there in the $(n + 1)^{st}$ position (which is the position right after the string S)? That value is the successor frequency of string S, in the corpus.

Harris believed that by calculating the successor frequency and the predecessor frequency at each point in each word of a corpus, he could find the morpheme boundaries (although Hafer and Weiss note that on the basis of their experiments neither choosing a threshold nor looking for a local maximum of successor frequency works very well in English). To make a long story short (see Goldsmith (2001, 2006) for the long version), such a purely local method does not work, and some more global characteristics of the overall grammar need to be taken into consideration, as we have already suggested.

Still, Harris's notion of successor frequency can serve as a useful heuristic for locating potential breaks, as the simple data in (1) suggest: the presence of the words *jump*, *jumped*, and *jumping* in a corpus leads to a successor frequency of three after the stem *jump*, just as it is after *walk*.

But successor frequency fails to work, even as a heuristic, when we turn to languages with much richer morphologies (that is, where the average number of morphemes per word is considerably higher than it is in English), and as linguists know, the morphological richness of English is on the poor side, as languages go.

The first case we will consider is that of the regular inflectional pattern of written modern French, which represents an earlier form of spoken French (some of this material is discussed in greater detail in Goldsmith and Hu (2004)). In the treatment of a subcorpus like *petit*, *petits*, *petite*, *petites*, *grand*, *grands*, *grande*, *grandes* (the masc. sg., masc. pl., fem. sg., and fem. pl. forms for *small*, *large*), the system we have described in Goldsmith (2006) will generate an FSA as in (11), and an algorithm described in Goldsmith and Hu (2004) generates the FSA in (12) rather than (13), which is the correct structure. The FSA in (11) misanalyzes the segmentation of the feminine plural forms, and (12) correctly segments, but does not represent the correct grammar, which is that given in (13). In terms of analogy, all three systems capture the analogy *petit* : *petits* : *petite* : *petites* :: *grand* : *grands* : *grande* : *grandes*, but only (13) expresses the analogy *petit* : *petits* :: *petite* : *petites* and also *petit* : *petite* :: *petits* : *petites*. (In fact, it appears to me easier to understand the nature of the generalization being captured by looking at the FSA than by using the traditional notation associated with analogy expressed with colons.)

$$\begin{tikzcd}
\bullet \arrow[r, bend left, "petit"] \arrow[r, bend right, "grand"'] & \bullet \arrow[r, bend left, "Ø"] \arrow[r, "s"] \arrow[r, bend right, "e"'] \arrow[r, bend right=70, "es"'] & \bullet
\end{tikzcd} \tag{11}$$

$$\begin{tikzcd}
\bullet \arrow[r, bend left, "petit"] \arrow[r, bend right, "grand"'] & \bullet \arrow[r, "e"] \arrow[rr, bend left=60, "Ø"] \arrow[rr, bend right=60, "s"'] & \bullet \arrow[r, bend left, "Ø"] \arrow[r, bend right, "s"'] & \bullet
\end{tikzcd} \tag{12}$$

$$\begin{tikzcd}
\bullet \arrow[r, bend left, "petit"] \arrow[r, bend right, "grand"'] & \bullet \arrow[r, bend left, "Ø"] \arrow[r, bend right, "e"'] & \bullet \arrow[r, bend left, "Ø"] \arrow[r, bend right, "s"'] & \bullet
\end{tikzcd} \tag{13}$$

The two big questions are: does a natural complexity measure unambiguously choose (13) over (11) and (12), and do we have a good search procedure that finds (13)? A relatively brief summary provides a positive answer to the first question; the second is more difficult to answer, and I will leave it open for now. The complexity of an FSA is almost exactly equal to the sum of the informational complexity associated with each of its nodes plus that of each of its edges plus that associated with the labels on the edges. As noted above, the informational complexity is in each case the inverse log probability of the item in question. In (11), there are three nodes, each of which has roughly the same informational complexity, equal in this case to $\sigma = -log\ S$, where S is the frequency of words that is described by this FSA in the corpus (that is, the total count of the words in this FSA divided by the total number of words in the corpus). The information complexity of the labels on each edge are also equal to the inverse log frequency of their usage, and *es* is a relatively rare suffix in French (i.e., there are relatively few feminine plural adjectives), and hence its informational cost is quite large. In addition, one must pay twice for the two pointers to each of the suffixes ø and *s*, and there is one more node in (12) than in (11). Hence (12) turns out to be more costly than (11). By contrast, (13) is less complex than either (11) or (12), despite the fact that it has one more node than (11). By avoiding positing a morpheme *es* (expensive because rare—it costs less than (11)), while by positing *s* only once, it costs less than (12).

I think this example clearly illustrates the basic point of this paper: formal complexity can, in many cases, be used to evaluate and compare alternative analyses, and algorithmic and information-theoretic complexity suffices to define the relevant complexity.

The second example we will look at represents still uncharted waters. It come from Swahili; consider (14), which gives a sample of some of the richness of the Swahili finite verb; I use the traditional Bantu terminology where appropriate. The positions indicated in this diagram illustrate subject markers, tense markers, object markers, verb roots, the passive/active marker, and the final vowel, respectively; there are also other affixes, such as a relative clause marker that can appear after the tense markers, which are not indicated here. There is little question but that the correct solution is formally much simpler than any of the partial solutions; algorithmic complexity will correctly identify an FSA as in (14) as a very simple grammar.

$$\begin{tikzcd}
\bullet \arrow[r, bend left=40, "ni"] \arrow[r, bend left=20, "u"] \arrow[r, "a"] \arrow[r, bend right=20, "tu"'] \arrow[r, bend right=40, "wa"'] & \bullet \arrow[r, bend left=40, "li"] \arrow[r, bend left=20, "ka"] \arrow[r, bend right=20, "na"'] \arrow[r, bend right=40, "ta"'] & \bullet \arrow[r, bend left=40, "ni"] \arrow[r, bend left=20, "ku"] \arrow[r, "m"] \arrow[r, bend right=20, "tu"'] \arrow[r, bend right=40, "wa"'] & \bullet \arrow[r, bend left=40, "imb"] \arrow[r, bend left=20, "pend"] \arrow[r, "fik"] \arrow[r, bend right=20, "som"'] \arrow[r, bend right=40, "chaku"'] & \bullet \arrow[r, bend left, "\emptyset"] \arrow[r, bend right, "w"'] & \bullet \arrow[r, "a"'] & \bullet
\end{tikzcd} \tag{14}$$

In order to even have a chance to discover these morphemes and the structure that lies behind them, we need to implement the notion of analogy in a richer fashion; what follows is taken from Hu *et al.* (2005).

We first look for elementary alignments between pairs of strings, as in (15), where m_1 or m_4 can be null, and m_2 or m_3 can be null. These elementary alignments can be found using the well-known string edit distance algorithm.

$$\begin{tikzcd}
\bullet \arrow[r, "m_1"'] & \bullet \arrow[r, bend left, "m_2"] \arrow[r, bend right, "m_3"'] & \bullet \arrow[r, "m_4"'] & \bullet
\end{tikzcd} \tag{15}$$

We expand these structures by finding ways to collapse them, either as suggested by (16), or as in (17) and (18).

$$\begin{tikzcd}
\bullet \arrow[r, "a"'] & \bullet \arrow[r, bend left, "li"] \arrow[r, bend right, "na"'] & \bullet \arrow[r, "yesema"'] & \bullet
\end{tikzcd} \tag{16}$$

$$\begin{tikzcd}\bullet \arrow[r, "a"'] & \bullet \arrow[r, bend left, "li"] \arrow[r, bend right, "na"'] & \bullet \arrow[r, "mfuata"'] & \bullet\end{tikzcd} \qquad (17)$$

$$\begin{tikzcd}\bullet \arrow[r, "a"'] & \bullet \arrow[r, bend left, "li"] \arrow[r, bend right, "na"'] & \bullet \arrow[r, bend left, "yesema"] \arrow[r, bend right, "mfuata"'] & \bullet\end{tikzcd} \qquad (18)$$

But establishing a clear and workable algorithm to correctly collapse these FSAs is no simple task, in the presence of only a realistic amount of data (and it is not clear that increasing the amount of data available would change the difficulty in an essential way). The simple cases illustrated here work fine to collapse small FSAs when the difference between them is small. But the problem becomes harder quite quickly when we try to induce the correct structure, for example, of what is perhaps the structure best represented in the data, that found in the first two "columns" of (14), representing the subject markers and the tense markers. Because each column has a large number of possible morphemes in it, the subgeneralizations that we easily find—typified by the one in (18), which has a single subject marker (*a*) followed by two tense markers (*li* and *na*)—become harder and harder to analogize to.

Let's be a bit more specific, to make concrete what we're talking about. In a corpus of 25,000 Swahili words (4,100 distinct words among them), we find 254 three-state FSAs with the methods we have sketched, and of these, virtually all of them are linguistically reasonable; the place where the strings are cut are, indeed, morpheme boundaries from the linguist's perspective. These three-state FSAs (and I have sketched the top eight in (19–26)) can be ranked with respect to how much information they compress: those that compress a good deal of information are necessarily those that express a large number of words with relatively few edges in the automaton. In theory, that kind of compression can happen in either of two ways: by specifying an FSA with a single stem but a wide range of affixes, or by specifying an FSA with a smaller set of affixes and a wide range of stems. It turns out that the latter is by far the most common kind of generalization obtained.

The task now is to generalize, which is effectively just another way of saying to learn what the morphological pattern of Swahili is. As far as I can see, there is little or nothing that we can posit as a simple innate premise that will help, nor will appealing to analogy help us, because the question now is really: when should two (or more) patterns be treated as analogous? Now, it is very likely true that if these strings of letters were labeled as the morphemes that they are (that is, if the labels told us more than just the phonemes: if they furthermore identified the functional category of the morpheme), our task would be considerably lightened. But taking that information for granted seems to me like question-begging. Swahili, just like most languages, often employs the same sequence of phonemes to realize different morphemes (for example, the subject and object markers for various person and number classes is the same: *tu* marks both subject and object marker for first-person plural, etc.) It is morphological analysis, and the inference of a morphological generator, that is an important step on the way to understanding the morphological identity of strings of letters (or phonemes); we risk circularity if we assume that knowledge of morpheme identity can serve as the basis of our knowledge of the morphological grammar. We would like to understand how a learner would generalize by recognizing the identity of the prefix *a* in patterns (19), (20), (22), (24);[16] but *a* is the most common phoneme and also the most common morpheme in the language, and occurs with several functions; mere phonological identity is simply *not* enough to lead the learner to treat all occurrences of *a* in the same way.

$$\left\{\begin{array}{ll} \textit{a} & \text{Subject} \\ \textit{wa} & \text{Markers} \end{array}\right\} \left\{\begin{array}{l} \text{55 stems:} \\ \textit{baki} \\ \textit{ende} \\ \textit{fanye} \\ \ldots \end{array}\right\} \qquad (19)$$

$$\left\{\begin{array}{ll} \textit{a} & \text{Subject} \\ \textit{m} & \text{Markers} \end{array}\right\} \left\{\begin{array}{l} \text{17 stems:} \\ \textit{cheni} \\ \textit{kaanguka} \\ \textit{kapoteza} \\ \ldots \end{array}\right\} \qquad (20)$$

$$\left\{\begin{array}{l} \text{17 stems:} \\ \textit{akaongez} \\ \textit{alifany} \\ \textit{ameja} \\ \ldots \end{array}\right\} \left\{\begin{array}{ll} \textit{NULL} & \text{active} \\ \textit{w} & \text{passive} \end{array}\right\} \qquad (21)$$

[16] And perhaps (25): the system posits *ana* as a prefix, and it is an inductive leap to treat this as the concatenation of *a* and *na* at this point.

$$\left\{\begin{array}{ll} \textit{a} & \text{Subject} \\ \textit{wa} & \text{Markers} \end{array}\right\}\left\{\begin{array}{l} \text{14 stems:} \\ \textit{changa} \\ \textit{heshimuni} \\ \textit{lilokataa} \\ \ldots \end{array}\right\} \tag{22}$$

$$\left\{\begin{array}{l} \text{12 stems:} \\ \textit{akawaachi} \\ \textit{amewaweke} \\ \textit{fany} \\ \ldots \end{array}\right\}\left\{\begin{array}{ll} \textit{a} & \text{default ending} \\ \textit{eni} & \text{plural imperative} \end{array}\right\} \tag{23}$$

$$\left\{\begin{array}{ll} \textit{a} & \text{Subject} \\ & \text{Marker} \end{array}\right\}\left\{\begin{array}{ll} \textit{li} & \text{Tense} \\ \textit{na} & \text{Markers} \end{array}\right\}\left\{\begin{array}{l} \text{11 stems:} \\ \textit{batiza} \\ \textit{chaguliwa} \\ \textit{kwenda} \\ \ldots \end{array}\right\} \tag{24}$$

$$\left\{\begin{array}{ll} \textit{ana} & \text{Subject Marker} \\ & \text{and Tense Marker} \end{array}\right\}\left\{\begin{array}{ll} \textit{NULL} & \text{default} \\ \textit{ye} & \text{Rel Clause marker} \end{array}\right\} \times \left\{\begin{array}{l} \text{10 stems:} \\ \textit{fanana} \\ \textit{ishi} \\ \textit{kuja} \\ \ldots \end{array}\right\} \tag{25}$$

$$\left\{\begin{array}{l} \text{18 stems:} \\ \textit{akili} \\ \textit{bahari} \\ \textit{dunia} \\ \ldots \end{array}\right\}\left\{\begin{array}{ll} \textit{NULL} & \text{default} \\ \textit{ni} & \text{postposition} \end{array}\right\} \tag{26}$$

We are currently working on a method to link the low-level FSAs illustrated in (19–26) to the larger, simpler, and correct pattern, that of (15), and I will sketch the intuition that lies behind it. These FSAs can be thought of themselves as expressions (for example, by alphabetizing all the elements in a column and concatenating them with a punctuation marker between them), and we can establish a distance measure across pairs of string expressions which we can then use to hypothesize which items should be collapsed to form a larger generalization. When two or more morphemes—especially high-frequency morphemes—appear in the same column (that is, in a paradigmatic morphological relationship), then they may be analyzed as likely alternatives for the same morphological position.

This is easier to explain with a real example. There are several high-frequency FSAs that begin with the subject marker *a*, followed by two alternative tense markers, followed by a set of verbal stems. In the first case, the two tense markers are *li* and *na*; in the second, the two tense markers are *li* and *me*; in the third, they are *ka* and *na*; in the fourth, *ka* and *li* (I have not listed these FSAs here). We can capitalize upon each of these pairings to create a distance metric among these morphemes with this information, increasing the simplicity of assigning them to the same morphological position. We do this in order to overcome the problem of the sparsity of the data: we never find a single stem in a finite corpus appearing in all of its possible forms; what we need to do is use the partial information that the data actually provide, and much of that information is bundled into the observation that various subsets of morphemes appear in the same position of the word—and we can infer that even before we have a clear global understanding of what the overall structure of the word is. In a sense, that's the key to understanding learning: understanding how we can incrementally advance the analysis of the data, through analyzing the data, even though we have not yet achieved a global understanding of how everything fits together. In this case, the appearance of a pair of stems (*keti, mtuma*) appearing with the subject marker *a* and three of the four tense markers (*ki, li, na*, in fact) strongly supports the hypothesis that they are all realizations of the same morphological position. The sense in which this is true can be mathematically formulated and integrated into the search algorithm. But considerable work remains if we are to correctly induce the simple, and globally coherent, morphological structure of forms like the Swahili verb.

7.8 Discussion and conclusion

We have covered—or at least touched on—quite a number of topics, all closely joined by the question of how morphology can be learned. We have focused on the task of learning to segment words into morphs and discovering the grammar which puts them back together. This task is already difficult enough, but I hope it is clear that in a sense this task is a surrogate for the larger and more difficult task of segmenting entire utterances (into the pieces we call words) and discovering the grammar which puts them back together.

In the case of morphology, there is little or no hope that an appeal to a magical slate of innate principles will greatly simplify the task (I refer, of course, to an information-rich Universal Grammar). As far as learning morphology is concerned, Locke was surely right: all the reasoning is search and casting about; it requires pains and application. But we must not lose sight of the fact that even if language learning means searching and casting about on the part of the learner, there still must be an overarching model which describes what it is that is being sought. It seems to me that only a highly mathematical model which comes to grips with the complexity (in the technical sense) of the hypothesis has even a chance of shedding light on the problem of language learning. And if this conjecture is correct, then it seems to me almost a certainty that the same learning mechanisms can be used to induce a syntax as well. While it is not logically impossible that learning morphology requires a rich and powerful learning theory and learning syntax does not, such a state of affairs is highly unlikely at best.

A word, in closing, is perhaps appropriate regarding the relationship between the kind of linguistic work we have sketched and the study of child language acquisition, since it is only natural to ask what connection is being posited between the two. The two answer different questions: the linguist asks how language *can be* learned; the psycholinguist asks how language *is* learned. Each has his work cut out for him. If the linguist had several adequate theories of how language could be learned, the psycholinguist could figure out which was the right one—but the linguist does not. If the psycholinguist could provide an account of how language *is* learned, we would have at least one answer to the question as to how language can be learned—but the psycholinguist does not. We are making progress, I think, regarding the models on the market for morphology learning, and some aspects of phonology learning, and there is a time-honored law according to which once we find *one* way to accomplish something, several more will present themselves virtually overnight.

These questions are reflections of an old and traditional debate between rationalist and empiricist inclinations in the study of mind, but the most familiar versions of how *both* schools have treated language acquisition are, in my view, coarse oversimplifications. Rationalists of the principles-and-parameters sort attempt to account for language learning by denying its existence, and hoping that the variation across the world's languages will simply go away, while empiricists of the old school hope that knowledge can be reduced to memory. Both of these are losing strategies, in my view, and I have tried to offer some specifics with regard to one small, but not insignificant, part of language learning. It is an empiricist account that sets a high bar for formal grammatical accounts of the relevant data.

7.9 Appendix

Let us consider how the probability of a corpus changes when we begin our word discovery process. Originally, the lexicon consists of the observed letters in the corpus. Our first guess will add the string TH to the lexicon. When we add the element TH, the log probability of the corpus is changed in three ways. First, the total number of words in the corpus decreases by the number of THs found in the corpus (that may not be obvious, but it is true, if you think about it). Second, the total number of Ts and Hs also decrease (since a T that is followed by an H is no longer parsed as a T, but rather as part of a TH), and hence the probability of both Ts and Hs decreases, since those probabilities are based on observed frequencies. (Note, by the way, that this illustrates the point that even frequencies are theory-dependent notions!) Third, the probability of the substring TH has gone up considerably, because it had previously been calculated as the product of the probabilities of T and H independently, but now it is calculated on the basis of the observed frequency of the sequence TH. The actual change in log frequency is $-N\Delta N + [t]\Delta[t] + [h]\Delta[h] + [th]\ log\frac{freq_2(th)}{freq_2(t)freq_2(h)}$, where N is the original length of the corpus and thus the number of words on the first analysis, ΔN is the log ratio of the count of words *after* versus *before*, i.e., $log\frac{N-number\ of\ THs}{number\ of\ letters}$, $[t]$ and $[h]$ are the number of *T*s and *H*s in the original corpus, $\Delta[t]$ is the log ratio of the counts of *T* after vs before and likewise for $\Delta[h]$, and $[th]$ is the number of substrings *TH* found in the corpus; $freq_2(x)$ is the frequency of x in the second model, that in which TH is interpreted as a single lexical item. Note that ΔN, $\Delta[t]$, and $\Delta[h]$ are all negative.