Segmentation and morphology

John A. Goldsmith[1]

Departments of Linguistics and Computer Science, The University of Chicago
goldsmith@uchicago.edu

A draft chapter for the Blackwell *Computational Linguistics and Natural Language Processing Handbook*, edited by Alex Clark, Chris Fox and Shalom Lappin. This draft formatted on 2nd May 2009.

Contents

1 Introduction

1.1 General remarks

The field of morphology has as its domain the study of what a word is in natural languages. In practice, this has meant the study of four relatively autonomous aspects of natural language: (1) the identification of the lexicon of a language, (2) morphophonology, (3) morphosyntax, and (4) morphological decomposition, or the study of word-internal structure.[1]

At first blush, identifying the lexicon of a language—what the words are—may seem simple, especially in languages which are conventionally written with spaces between words, but as we shall see below, the task is more complicated at more points than one would expect, and in some scientific contexts we may be interested in knowing under what conditions the spaces that mark separation between words can be predicted. To explain what points (2) through (4) above cover, we introduce the notion of *morph*—a natural, but not entirely uncontroversial notion. If we consider the written English words *jump, jumps, jumped*, and *jumping*, we note that they all begin with the string *jump*, and three of them are formed by following *jump* by *s, ed*, or *ing*. When words can be decomposed directly into such pieces, and when the pieces recur in a functionally regular way, we call those pieces *morphs*. With the concept of *morph* in hand, we may consider the following definitions:

- Morphophonology. It is often the case that two or more morphs are similar in form, play a nearly identical role in the language, and can each be analytically understood as the realization of a single abstract element—"abstract" in the sense that it characterizes a particular grammatical function, and abstracts away from one or more changes in spelling or pronunciation. For example, the regular way in which nouns form a plural in English is with a suffixal *-s*, but words ending in *s, sh*, and *ch* form their plurals with a suffixal *-es*. Both *-s* and *-es* are thus morphs in English, and we may consider them as forming a class which we call a *morpheme*: the pair of morphs {*s, -es*}, whose grammatical function is to mark plural nouns. The principles that are involved in determining which morph is used as the correct realization of a morpheme in any given case is the responsibility

[1] I am endebted to many friends and colleagues, both for conversations on the topics discussed in this paper and for comments on an earlier draft, including Carl de Marcken, Paul Cohen, Walter Daelemans, Tomaž Erjavec, Antonio Galves, Sharon Goldwater, Yu Hu, Mark Johnson, Chunyu Kit, Ursula Klenk, Kimmo Koskenniemi, Colin Sprague, Richard Sproat, Antal van den Bosch, J. G. Wolff, Aris Xanthos, and the editors of this volume; I hope that I have succeeded in correcting the errors they pointed out to me. Reader, bear in mind that all the ideas presented here have been simplified to improve comprehensibility. As always, if you are interested, read the original.

of morphophonology. Morphophonology is the shared responsibility of the disciplines of phonology and morphology.

- Morphosyntax. Syntax is the domain of language analysis responsible for the analysis of sentence formation, given an account of the words of a language.[2] In the very simplest cases, the syntactic structure of well-formed sentences in a language can be described in terms of atomic and unanalyzed words, but grammar is never really that simple. In reality, the morphs that appear inside one word may also specify information about other words in the sentence—for example, the verbal suffix *-s* in *Sincerity frightens John* specifies that the subject of the verb is grammatically singular. Thus statements about syntax inevitably include some that peer into the internal structure of at least some words in a language, and in many languages this is the rule rather than the exception. Morphosyntax deals with the relationship between the morphemes found inside one word and the other words that surround it in the larger sentence; it is the shared responsibility of the disciplines of syntax and morphology.
- Morphological decomposition. While English has many words which contain only a single morpheme (e.g., *while, class, change*), it also has many words that are decomposable into morphs, with one or more suffixes (*help-ful*, *thought-less-ness*), one or more prefixes (*out-last*) or combinations (*un-help-ful*). But English is rather on the tame side as natural languages go; many languages regularly have several affixes in their nouns, adjectives, and even more often, their verbs (e.g., Spanish *bon-it-a-s*, which consists of a root meaning **good**, a diminutive suffix *-it*, a feminine suffix *-a*, and a plural suffix *-s*).

In the remainder of this introductory section, we will give a brief overview of the kinds of questions that have traditionally been the focus of the study of morphology in general linguistics. This will serve as background to the discussion of the following three questions which are specifically computational in character.

(1) Can we develop—and if so, how—a *language-independent* algorithm that takes as input a large sequence of symbols representing letters or phonemes and provides as output that same sequence with an indication of how the sequence is divided into words? This question puts into algorithmic form the question of how we divide a string of symbols into words.
(2) How can we develop a language-independent algorithm that takes as input a list of words and provides as output a segmentation of the words into morphemes, appropriately labeled as prefix, stem, or suffix—in sum, a basic morphology of the language that produced the word list?
(3) How can we implement our knowledge of morphology in computational systems in order to improve performance in natural language processing?

[2] See chapters 4,10, and 15 in this book.

1.2 Morphology

Users of natural languages (which is to say, all of us) need no persuasion that words are naturally occurring units. We may quibble as to whether expressions like *of course* should be treated as one word or two, but there is no disagreement about the notion that sentences can be analytically broken down into component words. Linguists and others who think deeply about such questions as "what is a *word*?" generally focus on the idea that there is evidence in each language from phonology, morphology, syntax, and semantics which points in the direction of a natural *chunk* corresponding to the traditional notion of word. From a phonological point of view, phenomena that occur inside a word are often quite distinct from phenomena that occur at word-boundary—the conditions under which a *t* is a flap in American English differ considerably in this way, for example. In a similar way, we find that at the point in an utterance between two words, we can expand the utterance by adding additional material. For example, we can convince ourselves that *their* is a separate word, and not a prefix to the word *dream*, because we can say: *John and his wife will follow their—or at least his—dream next year.*

There are some difficult intermediate cases which linguists call *clitics*, morphemes whose status as a full-fledged word is dubious; the possessive suffix *′s* in English is such a case, because although in many respects it seems like a suffix to the word that precedes it, it may nonetheless be syntactically and semantically associated with a preceding phrase, as in an example like *a friend of mine's first husband* (contrast this with *a friend of my first husband*).

In all languages, or virtually all, it is appropriate to analytically break words down into component pieces, called *morphs*, and then to bundle morphs back into the functional units we call morphemes; such an analysis is part of the functionality of a morphology, and is the central subject of this chapter (when a morpheme corresponds to only a single morph, as is often the case, we generally ignore the difference between a morph and a morpheme). In addition, we expect of a complete morphology that it will associate the appropriate set of morphosyntactic features with a word, to the extent that the word's morphological decomposition can serve as a basis of specifying those features. Thus *books* should be analyzed as *book* plus a suffix *s*, and the suffix *s* should be marked as indicating plurality for nouns in English.

Morphologies are motivated by four considerations: (1) the discovery of regularities and redundancies in the lexicon of a language (such as the pattern in *walk:walks:walking :: jump:jumps:jumping*); (2) the need to make explicit the relationship between grammatical features (such as nominal NUMBER or verbal TENSE) and the affixes whose function it is to express these features; (3) the need to predict the occurrences of words not found in a training corpus; and (4) the usefulness of breaking words into parts in order to achieve better models for statistical translation, information retrieval, and other tasks that are sensitive to the meaning of a text.

Thus morphological models offer a level of segmentation that is typically larger than the individual *letter*,[3] and smaller than the *word*. For example, the English word *unhelpful* can be analyzed as a single word, as a sequence of nine letters, or from a morphological point of view as a sequence of the prefix *un*, the stem *help*, and the suffix *ful*.

The distinction between *inflectional* morphology and *derivational* morphology is one drawn by most accounts of morphology, but it remains a controversial question for some as to whether a clear line can be drawn between the two. The intuition that lies behind the distinction is reasonable enough; to illustrate this, let us consider an example from English. We may wish to say that *jump*, *jumps*, and *jumped* are three words, but they are all different versions (in some sense) of a single verb stem. The verb stem (*jump*) is coherent in three ways: it has a recognizable phonological form (*jump*), it shares a coherent semantic content, and it is *inflected* in ways that it shares with many other stems: in particular, it takes a suffixal *-s* in the third person singular present tense, and an *-ed* in the past tense. In addition to the characteristics just mentioned, inflectional affixes also are usually *peripheral*—if they are suffixes, the inflectional suffixes are at the very end of the word, and if prefixes, at the very beginning, and while they contribute grammatical information to the word they contain, they do not shift the part of speech of their word. The suffixes *-s, -ed* are taken to be inflectional suffixes, and they differ from derivational suffixes such as *-ity* (as in *sanity*) or *-ness* (as in *goodness, truthiness*[4]) or *-ize* (as in *radicalize, winterize*). Derivational affixes more often than not play the role of indicating a change of part of speech, in the sense that *sane* is an adjective, and *san-ity* is a noun, just as *radicalize* and *winterize* are verbs, but contain within them stems of a different category (adjective and noun, respectively). In addition, the semantic relationship between pairs of words related by derivational affixes is often far less regular than that found between pairs of words related by inflectional affixes. Thus, while the relationship between *jump* and *jumped*, *walk* and *walked*, and so on, is semantically regular, the same cannot be said of the relationship between words such as *woman* and *womanize*, *author* and *authorize*, and *winter* and *winterize*.[5]

For all of these reasons, most accounts of morphology distinguish between the analysis of a word's inflectional morphology, which isolates a stem (an *inflectional stem*) from its inflectional affixes, and the word's derivational morphology, which further breaks the (inflectional) stem into component pieces.

[3] Since computational linguists have traditionally interested themselves more with written language than spoken language, I write here of *letters* rather than *phonemes*, but the reader who is interested in spoken language should substitute *phoneme* for *letter* in the text.

[4] http://en.wikipedia.org/wiki/Truthiness

[5] There are two suffixes *-ing* in English; the one that appears in sentences in the progressive (*John is running*) is inflectional; the one that creates nominals is derivational (*No singing of songs will be tolerated.*)

Thus *winterized* is analyzed into a stem *winterize* plus an inflectional suffix *ed*, and the stem *winterize* is divided into a stem *winter* plus a derivational suffix *ize*. The term *root* is often used to refer to a stem that cannot be morphologically decomposed. An inflectional stem is associated with a single lexical category, such as noun, verb, adjective, etc. Just as importantly, the inflectional stem is the item in a lexicon which can be (and usually is) associated with a particular meaning, one that is generally not strictly predictable from the meanings of its components.

It is not unusual to find situations in which (by what I have just said) we find two stems that are spelled and pronounced identically: *walk* is both a noun and a verb, for example. The term *conversion* is often used to refer to this situation, in which a stem is (so to speak) converted from one part of speech to another without any overt affixation, though such analysis generally assumes that one can determine which of the two (noun or verb, in this case) is the more fundamental category of the two, on a stem by stem basis.

Inflectional stems can also be created by a process of compounding, as in *watchdog* or *eyepatch*. Such compound stems may include inflectional affixes, though many languages impose rather severe restrictions on the inflectional morphology permitted inside a compound (this is distinct from the case in which inflectional affixes "attach" to compounds, as in *watchdog-s*, and exceptions exist, such as *months-long*, which is quite different in meaning from *month-long*). In some languages, a compound is formed by concatenating two stems; in others, a short linking element appears between them. The linking element of Greek compounds, *-o-*, appears in many compounds borrowed into English, such as in *hipp-o-potamus*.

All of the broad generalizations that I have suggested to this point, like most such broad generalizations, only go so far, and there are always phenomena in a natural language which demand a more complex view. I will sketch here some of the ways in which complexities arise most often.

First of all, in inflectional systems, there are typically a set of anywhere from two to a dozen relatively independent grammatical features which may be relevant to a particular word class, such as noun or verb. For example, a verb may be specified for the PERSON and NUMBER of its subject, of its object, and for its TENSE, and for other characteristics as well. Only rarely—indeed, vanishingly rarely—is each such feature realized separately as its own morph. In most cases, it is a small tuple of features that is linked to a particular affix, as in the case of the English verbal suffix *-s*, which marks 3rd person & singular & present-tense. On the other hand, it is often the case that a single affix is used to mark more than one tuple of features; in written French, the suffix *-is* marks the present tense singular subject agreement marker for a certain class of verbs; for example, *finis* is either '*(I) finish*' or '*(you (sg.)) finish*', in either the 1st or 2nd person, but not in the 3rd person(which is spelled *finit*).

For this reason, linguists often think of inflectional systems as being hyper-rectangles in a large space, where each dimension corresponds to a grammatical feature, and where the edge of a hyper-rectangle is divided into intervals

corresponding to the feature values that the feature may take on (that is, *person* is a feature, and it may take on the values 1*st*, 2*nd*, or 3*rd* ; NUMBER is a feature, and it may take on the values singular and plural, in some languages). Each affix will be associated with one or, quite often, several small sub-hyperrectangles in such a system.

The complexities do not stop there. It is often the case that there are two or more forms of the stem used, depending on which subpart of the inflectional hyper-rectangle we are interested in. An extreme case is that of the stem *went* in English, used as the stem in the past, when *go* is used otherwise. This case is a bit special, since the form of *go* and *went* is so different (when the stems are this different, linguists refer to this as *suppletion*), but it is often found that several related (but distinct) stems will be used for different parts of the system. In French, for example, the present tense stem for '*write*' is spelled *'écri-'* in the singular, but *'écriv'* in the plural. This is often referred to as *stem alternation*.

In addition, a language may employ a whole arsenal of different inflectional hyper-rectangles, even within a single lexical category. The Romance languages are perfectly typical in having between three and six so-called 'verb classes,' which employ quite different sets of suffixal patterns for marking precisely the same set of grammatical features. It is the verb stem which decides which inflectional set will be used for its morphology. See Goldsmith & O'Brien (2006).

Finally, we must acknowledge that not all morphology is properly thought of as the concatenation of morphs. In English, and many of the other Indo-European languages, we find inflectional patterns on verbs which consist of sets of stems (these are called *strong verbs*) that differ primarily with regard to the vowel: the past of *stand* is *stood*, the past of *sing* is *sang*, and the past of *catch* is *caught*. We will focus on those aspects of morphology which are strictly concatenative—in which words can be analyzed as sequences of morphs—but we will return to the treatment of the more general case below as well.

1.3 Static and dynamic metaphors

Inflectional morphology is complex in most natural languages. It is common for nouns to be marked morphologically for number and case, and for verbs to be marked morphologically for tense, person, number, mood (whether the verb is in the indicative or the subjunctive), and syntactic position (whether the verb is in a subordinate clause of the sentence or not), for example. In fact, the hallmark of inflectional morphology—how we recognize it when we see it—is the appearance of several features that are logically orthogonal to one another, all of which are relevant for the realization of all, or most, of the words in a given part of speech (noun, verb, adjective). To put that a bit more concretely: to know Latin morphology is to know that a given verb is specified for the features of person, number, tense and mood. The verb *cantō* is in the

FIRST PERSON, SINGULAR, PRESENT TENSE INDICATIVE form, and the same is true of a very large, and potentially unbounded, set of verbs ending in -*ō*.

Lying behind that very specific knowledge is the understanding that FIRST PERSON is a value of the feature person, that SINGULAR is a value of the feature number, that PRESENT is a value of the feature tense, and that INDICATIVE is a value of the feature mood. There are some dependencies among these features: there are more tenses when the mood is indicative and fewer when it is subjunctive, but these dependencies are the exception rather than the rule among inflectional features. There is no logical ordering of the features, for the most part: there is no logical or grammatical reason for person to precede number, or to follow it (there may well be linear ordering of the morphemes that realize these morphemes, though.) For all of these reasons, inflectional systems can encode many different combinations of feature specifications—quite unlike what we find with derivational morphology.

Some regularities in the morphology of a language are best expressed in terms that refer only to these inflectional features: for example, while the feature tense in Spanish may take on four values in the indicative mood (PRESENT, FUTURE, AORIST, and IMPERFECTIVE), it takes on only two values in the subjunctive mood (PRESENT and PAST); in German, the forms of the NOMINATIVE and ACCUSATIVE are the same for all neuter nouns. On the other hand, other generalizations address characteristics of the phonological realization of these features as morphs: in Finnish nouns, PLURAL number is marked (temporally, graphically) before case.

Much of contemporary linguistic theory is dominated in a curious way by the belief that there is a correct order in which various aspects of the representation of a word or sentence is *constructed*; derivational theories are the clearest example of this. Reflecting on this, Stump 2001 distinguishes between *incremental* approaches, in which the process of adding an affix also adds morphosyntactic features, and *realizational* approaches, in which the process of adding an affix has access to a representation in which morphosyntactic features are present (or *already* present, as a derivationalist would have it). However, it is frequently the case that the distinction between these two approaches vanishes in a computational implementation, either because the analysis is conceptually static rather than dynamic (that is, it places well-formedness conditions on representations rather than offering a step-by-step method of producing representations), or because the dynamic that the computational implementation embodies is a different one. See Roark & Sproat (2006), chapter 3, for a detailed discussion of this point in the context of finite state transducers.

All of the material presented in this section is the result of the work of generations of linguists reflecting on many languages. In the next two sections, we will consider how the problem of *learning* about words and morphological structure can be reconstructed as a computational question of learning.

2 Unsupervised learning of words

In this section, we will discuss the computational problem of discovering words from a large sequence of symbols that bear no explicit indication of where one word ends and the next begins. We will start by distinguishing two formulations of this problem: one relatively easy, and the other quite difficult.

2.1 The two problems of word segmentation

Let us consider strings of symbols chosen from an alphabet Σ, which the reader may think of as the letters of a written language or the sounds of a spoken language. There are two broad families of ways in which we analyze the structure of strings of symbols. One uses probabilistic models, which tell us about the probabilities of selection of elements from Σ in the future, given the past, typically the very local, recent past. In such models, the structure that we impose lies in the departure of the system from a uniform distribution, and the probability of a symbol is typically conditioned by a small number of immediately preceding symbols. The other uses segmentation models, whose purpose is to allow for the restructuring of a string of elements from a fine-grained alphabet (such as Σ) to a coarser set $\mathcal{L}$ which we call a *lexicon*, and which should be thought of intuitively as a set of substrings generated from Σ, that is, as a subset of Σ^*. For now, we may simply think of the members of $\mathcal{L}$ as our *words*. We will focus primarily on the second family of models, those employing *chunking*, but I do not wish to even suggest that there is any sort of incompatibility between the two approaches, because there is not.

Each word $w \in \mathcal{L}$ is associated with an element of Σ^*, its *spell-out*—I write "associated with" rather than "is," because w may be decorated with other information, including meaning, syntactic category, and so on; but for simplicity of exposition, we may assume that no two elements in a lexicon are associated with the same spell-out. $\mathcal{L}^*$ is any concatenation of words, and any member s of $\mathcal{L}^*$ has a natural way of being thought of as a member of Σ^*: any sequence of words is naturally thought of as a sequence of letters, too. So far, no delimiters, like space or other punctuation, have come into the picture.

We will always assume that each member of Σ is also a member of $\mathcal{L}$ (roughly speaking, each member of the alphabet is a word), and so we can be sure that any string in Σ^* corresponds to at least one member of $\mathcal{L}^*$, but in most cases that we care about, a string in Σ^* will correspond to more than one string in $\mathcal{L}^*$, which is just a way of saying that breaking a string into words is not trivial. Each member of $\mathcal{L}^*$ which corresponds to a given string of letters we call a *parse* of that string. The string *atone* has three natural non-trivial parses: *atone*, *at one*, and *a tone*, but it has others as well.

The *first* problem of word-segmentation, then, is to find a method to take a string that in fact consists of strings of words, but which is presented as a string of letters with no indication of where one word ends and the next begins, and then from this string to reconstruct where the word breaks are.

It is easy to go from such a corpus C_1 in which words are separated by spaces to a corpus C_2 in which all spaces have been removed, but can we reconstruct C_1 from C_2 with no information beyond word frequency? Given a corpus C_1 which indicates word boundaries separating words, we can easily construct a lexicon L, and a new corpus C_2 in which the word boundaries have been eliminated. Can we find a language-independent algorithm $\mathcal{S}_1(L, C_2)$ that can reconstruct C_1? Put another way, can we define a method (either fool-proof, or just very good) that is able to put spaces back into a text with no more than a knowledge of the lexicon of the language from which the text was drawn?

There is no guarantee that such an algorithm exists for a specific language or for languages in general, nor is there a guarantee that if it exists (for a language, or for languages) that we can find it. Two families of natural approaches exist, the greedy and the probabilistic. The greedy approach scans through the string S: at position i, it looks for the longest substring s^* in S beginning at point i that appears in the lexicon; it then decides that s^* appears there in S, and it then skips to position $i+|s^*|$ and repeats the operation. The probabilistic model assumes a Markov probabilistic model over $\mathcal{L}^*$ (typically a 0 order or 1st order Markov model), and finds the string in $\mathcal{L}^*$ with the highest probability among all such strings whose spell-out is S.

In general, we may wish to develop an algorithm that assigns a probability distribution over possible analyses, allowing for ranking of analyses: given a string *anicecream*, we may develop an algorithm that prefers *an ice cream* to *a nice cream* by assigning a higher probability to *an ice cream* . Linguists working on Chinese and Japanese have contributed significantly to improvements in our understanding of this problem (see, e.g., Sproat *et al.* (1996), Ando & Lee (2003), Teahan *et al.* (2000) and the series of Chinese word segmentation bakeoffs easily found on the internet).

The *second* problem of word-segmentation is one large step harder than the first: given a long string of symbols with no breaks indicated, can we infer what the words are? See Figure 1. This problem asks whether it is possible find a general algorithm S_2 which takes as input a corpus C_2, which was created by stripping boundaries from a corpus C_1, and which gives as output a lexicon $\mathcal{L}$ which will satisfy the conditions for the lexicon L needed for S_1, the solution to the first problem.

Since for any large corpus which has been stripped of its word boundaries, there are an astronomical number of different lexicons that are logically consistent with that stripped corpus, it should go without saying that if we can solve the second problem for naturally occurring corpora in real languages, we do not expect it to be extendable to just *any* randomly generable corpus: to put it another way, to the extent that we can solve this problem, it will be by inferring something about the nature of the device that generated the data in the first place—something about the nature of human language, if it is natural language that we are exploring.

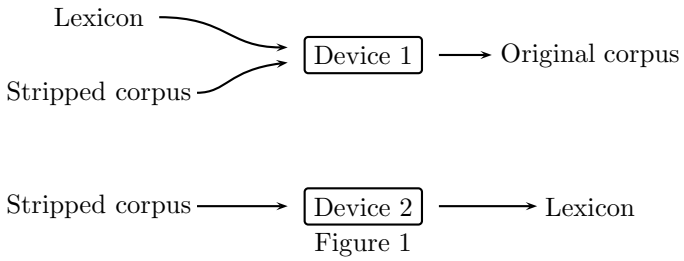


Figure 1

The problem of word segmentation may seem artificial from the point of view of someone familiar with reading Western languages: it is the problem of locating the breaks between words in a corpus. In written English, as in many other written languages, the problem is trivial, since we effortlessly mark those breaks with white space. But the problem is not at all trivial in the case of a number of Asian languages, including Chinese and Japanese, where the white space convention is not followed, and the problem is not at all trivial from the point of view of continuous speech recognition, or that of the scientific problem of understanding how infants, still incapable of reading, are able to infer the existence of words in the speech they hear around them.

Another computational perspective from which the problem of word breaking is interesting is this: to what extent do methods of analysis that have worked well in non-linguistic domains work well to solve this particular problem? This question is of general interest to the computer scientist, who is interested in a general way of regarding the range of problems for which an approach is suitable, and of considerable interest to the linguist, for the following reason. The most important contribution to linguistics of the work of Noam Chomsky since the mid 1950s has been his insistence that some aspects of the structure of natural language are unlearnable, or at the very least unlearned, and that therefore the specification of a human's knowledge of language *prior* to any exposure to linguistic data is a valid and an important task for linguistics. But knowledge of the lexicon of a given language, or the analysis of the words of the lexicon into morphemes, is a most unlikely candidate for any kind of innate knowledge. Few would seriously entertain the idea that our knowledge of the words of this book are matters of innate knowledge or linguistic theory; at best—and this is plausible—the linguist must attempt to shed light on the *process* by which the language learner infers the lexicon, given sufficient data. To say that the *ability* to derive the lexicon from the data is innate is something with which few would disagree, and to the extent that a careful study of what it takes to infer a lexicon or a morphology from data provides evidence of an effective statistically-based method of language learning, such work sheds important light on quite general questions of linguistic theory.

The idea of segmenting a long string $S \in \Sigma^*$ into words is based on a simple intuition: that between two extreme analyses, there must be a happy medium that is optimal. The two extremes I refer to are the two "trivial" ways to slice S into pieces: the first is to not slice it at all, and to treat it as composed of exactly one piece, identical to the original S, while the second is to slice it into many, many pieces, each of which is one symbol in length. The first is too coarse, and the second is too fine, for any long string that comes from natural languages (in fact, for most systems generating strings that are symbolic in any sense). The first explains too much, in a sense, and overfits the data; the second explains too little. The challenge is to find an intermediate level of chunking at which interesting structure emerges, and at which the average length of the chunks is greater than 1, but not enormously greater than 1. The goal is to find the right intermediate level—and to understand what "right" means in such a context. We will have succeeded when we can show that the string S is the concatenation of a sequence of members of a lexicon.

2.2 Trawling for chunks

There is a considerable literature on the task of discovering the words of an unlabeled stream of symbols, and we shall look at the basic ideas behind four major approaches.

Olivier

The first explicit computational model of word learning is found in Olivier (1968) (my description of this unpublished work is based primarily on Kit (2000)). The algorithm begins with a division of the corpus, in some arbitrary fashion, into successive chapters which will be analyzed, one at a time and successively. Letting i be 1, we establish an "provisional" lexicon from chapter i; it could be simply the lexicon consisting of all the individual letters of the alphabet that is used, and in some provisional fashion a probability distribution is established over the lexicon. Given that lexicon at pass i, we can relatively easily find the division of the chapter into words that maximizes the probability of the string, on the assumption that the probability of a string is simply the product of the (unigram) probabilities of its words. This maximum likelihood parse provides us with a new set of *counts* of the items in the lexicon, and normalizing, we take these as forming a new probability distribution over the lexicon. We now modify the lexicon by adding and removing some of its members. If we find that there is a sequence of two lexicon members that occurs more frequently than we would expect (by virtue of its frequency being greater than the product of the frequencies of its individual members), then we add that word to the lexicon. On the other hand, if a lexicon member occurs only once, we remove it from the lexicon. Having done this, we increment i, and reapply this process to the next chapter in the corpus.

This algorithm contains several elements that would be retained in later approaches to the problem, but in retrospect, we can see that its primary weakness is that it does not offer a principled answer to the question as to how large (i.e., how long) the words should be. It avoids the question in a sense by not re-applying recursively on a single text (chapter), but does not address the question head-on.

MK10

Another early lexicon-building approach was MK10, proposed by Wolff (1975, 1977). The initial state of the device is a lexicon consisting of all of the letters of the corpus. The iterative step is a continuous scanning through the text, parsing the text C at point i (as i goes from 1 to $|C|$) by finding the longest string s in the lexicon which matches the text from point i to point $i+|s-1|$, and then proceeding from the next point, point $i+|s|$; when i reaches the end of the corpus, we begin the scan again. The algorithm keeps track of each *pair* of adjacent "lexical items" that occur, and when the count of a particular pair exceeds a threshold (such as 10), the pair is added to the lexicon, all counts are reset to zero, and the process begins again. In sum, this is a greedy system that infers that any sequence which occurs at least 10 times is a single lexical item, or a part of a larger one.

Wolff's paper includes a brief discussion in which the relevance of his analysis is explicitly made to a number of important elements, including associationist psychology, the elimination of redundancy, natural selection, economy in storage and retrieval, induction, analysis by synthesis, probability matching, and the possibility of extending the algorithm to the discovery of lexical classes based on neighborhood-based distribution.

Sequitur

Craig Nevill-Manning, along with Ian Witten (see Nevill-Manning & Witten (1997); Nevill-Manning (1996)) developed an intriguing non-probabilistic approach to the discovery of hierarchical structure, dubbed *Sequitur*. They propose a style of analysis for a string S, employing context-free phrase-structure rules $\{R_i\}$ that are subject to two restrictions demanding a strong form of non-redundancy: (1) *no pair* of symbols S, T, in a given order, may appear twice in the set of rules, and (2) every rule must be used more than once. Violation of either principle gives rise immediately either to the creation of a new rule (if the first is violated) or to the elimination of a rule (if the second is violated). Such sets of rules can be viewed as compressions of the original data which reveal redundancies in the data. An example will make clear how this is done.

Suppose the data is *thecatinthehatisnocat*. The algorithm will begin with a single rule expanding the root symbol S as the first symbol, here t: $S \rightarrow t$. As we scan the next letter, we extend the rule to $S \rightarrow th$, and then to $S \rightarrow the$,

and so on, eventually to $S \rightarrow thecatinth$. Now a violation of the first principle has occurred, because *th* occurs twice, and the repair strategy invoked is the creating of a non-terminal symbol (we choose to label it 'A') which expands to the twice-used string: $A \rightarrow th$, which allows us to rewrite our top rule as $S \rightarrow AecatinA$, which no longer violates the principles. We continue scanning and extended the top rule now to: $S \rightarrow AecatinAe$, still maintaining the second rule, $A \rightarrow th$. Since *Ae* appears twice, we create a new rule, $B \rightarrow Ae$, and our top rule becomes: $S \rightarrow BcatinB$. But now rule A only appears once, in the rule expanding B, so we must dispense with it, and bulk up B so that it becomes: $B \rightarrow the$. We now see successful word recognition for the first time. We continue 3 more iterations, till we have $S \rightarrow BcatinBhat$. Since *at* is repeated, we create a rule for it $C \rightarrow at$, and the top rule becomes $S \rightarrow BcCinBhC$. Scanning 6 more times, we arrive at $S \rightarrow BcCinBhCisnocC$, after the final *at* is replaced by C. We then create a new rule $D \rightarrow cC$, leading to the top rule $S \rightarrow BDinBhCisnoD$. Here we stop, and we have the top level parse corresponding to $the-cat-in-the-h-at-isno-cat$, with groups corresponding to *the* and to $c-at$. As this example illustrates, the very strict conditions set on the relationship between the rule set and the compression representation of the data lead to a powerful method of extracting local string regularities in the data.

Sequitur is an instance of what has come to be known as grammar-based compression (Kieffer & En hui Yang (2000)), whose goal is to develop a formal grammar that generates exactly one string: the text being compressed, and the grammar itself serves as a lossless compression of the text. That there should be a logical connection between an optimal compression of a string of symbols and the structure that inheres in the system that generated the string lies at the heart of the next perspective we discuss, Minimum Description Length analysis.

MDL approaches

Some of the most interesting of the probabilistic approaches to word segmentation employ probabilistic models that are influenced by Kolmogorov's notions of complexity, such as Rissanen's notion of Minimum Description Length analysis. [6] These approaches provide explicit formulations of the idea mentioned above that word segmentation is a problem of finding a happy medium,

[6] For general discussion of this approach, see Rissanen (2007), Li & Vitányi (1993). The first general exploration of these ideas with application to linguistic questions was undertaken in Ellison (1994), and several developments along similar lines appeared in the early to mid 1990s, notably Rissanen & Ristad (1994), Cartwright & Brent (1994), Brent *et al.* (1995), Brent & Cartwright (1996), de Marcken (1996), Cairns *et al.* (1997), Brent (1999) and Kit & Wilks (1999) as well as Kit (2000). My discussion here presents a simplified version of the approach that is common to those accounts.

somewhere between the two extremal analyses of a text string: one extreme in which the string is treated as a single, unanalyzed chunk, and the other in which it is treated as a concatenation of single symbols, each a "chunk" separate from the previous one and the next one.

In a sense, the problem of discovering what the words are of a text means giving up any interest in what the specific message is that is encoded there, at least for the time being. If that sounds paradoxical, just think about it: in asking what the words are in an utterance and nothing else, we care about the building blocks of the message, not the message itself. So a hypothesis about the correct segmentation of a text is, in part, a hypothesis about what information is in the *message* being encoded, and what information is part of the larger system being used to perform the encoding—which is to say, the language; it is a hypothesis about the factorization of linear information into system and message. If we say that the entire text (*A Tale of Two Cities*, by Charles Dickens, for example) is a single lexical item, then we have truly missed the generalization that the work actually shares a lexicon with any other text in English! If we say that the lexicon of the text is simply the 26 or so letters needed to write it out, then we have also missed the generalization that there are many often repeated strings of symbols, like *it, times*, and *Paris*.

The heart of the MDL approach is the realization that each of those two extremes results in an overloading of one of two encodings. The first approach mentioned above, treating the text as a single lexical item, leads to the overloading of the lexicon; although it contains only one item, that single item is very, very long. The second approach leads to a single lexicon, with no more than a few dozen symbols in it, but specifying what makes *A Tale of Two Cities* different from any text which is not *A Tale of Two Cities* requires specifying every single successive letter in the text. That is simply too many specifications: there are far better ways to *encode* the content of the text than by specifying each successive letter. The better ways are ones in which there is a lexicon with the real words of the language, and then a spelling out of the text by means of that lexicon. Because the average length of the words in the lexicon is much greater than 1, the description of the text by specifying each *word*, one after the other, will take up much less space (or technically, far fewer bits of information) than specifying each letter, one after the other. The happy medium, then, is the analysis which minimizes the sum of these two complexities: the length of the lexicon and the length of the description of the text on the basis of that lexicon.

It turns out (though this is by no means obvious) that if we make our lexicon probabilistic, it is easy to measure the number of bits it takes to describe a specific text S, given a particular lexicon; that number is the negative base two logarithm of the probability of the text, as assigned by the lexicon (rounded up to the nearest integer); we write this $\lceil -log_2\, pr(S) \rceil$. Probability plays a role here that is based entirely on *encoding*, and not on *randomness* (i.e., the presumption of the lack of structure) in the everyday sense. Making the lexicon probabilistic here means imposing the requirement that it assign

a probability distribution over the words that comprise it, and that the probability of a word string S is the product of the probability of its component words (times a probability that the string is of the length that it actually happens to be). $\lceil -log_2\, pr(S) \rceil$ specifies exactly how many bits it would take to encode that particular message, using the lexicon in question. This is not obvious, but it is true.

How do we measure the number of bits in the description of the lexicon? A reasonable first approximation would be to calculate how much information it takes to specify a list of words, written in the usual way from an alphabet of a particular size. Ignoring a number of niceties, the length of a list of N words, each of length $|w_i|$, written in an alphabet consisting of m letters, is $log_2 N + \sum_{i=1}^{N} |w_i|\ log_2\ m$, which is very close to the length of the lexicon (times a small constant), that is to say, the sum of the lengths of the words that make up the lexicon. This quantity, c_L, is naturally referred to as the complexity, or information content, of the lexicon. We will come back to this, and consider some alternatives which make the description shorter; what I have just mentioned is a simple base-line for length, a length that we know we can easily calculate. In a manner entirely parallel to what I alluded to in the preceding paragraph, there is a natural way to assign a well-formed probability to a lexicon as well, based on its complexity c_L: it is 2^{-c_L}.[7]

Minimum Description Length analysis proposes that if we choose to analyze a string S into words (chunks that do not overlap, but cover the entire string), then the optimal analysis of S is by means of the lexicon L for which the sum of the two quantities we have just discussed forms a minimum: the first quantity is -$log_2\, prob\, S$ computed using given L, and the second quantity is C_L, which is the number of bits in description of lexicon L.

We can now turn all of this discussion of description length into an algorithm for the discovery of the words of a corpus *if* we can find a method for actually *finding* the lexicon that minimizes the combined description length.[8] A number of methods have been explored exploiting the observation that as we build *up* a lexicon from small pieces (starting with the individual letters [Line 1]) to larger pieces, the only candidates we ever need to consider are pairs of items that occur next to each other somewhere in the string (and

[7] The reason for this is that the quantity C_L essentially counts the number of 0s and 1s that would be required to efficiently express the lexicon in a purely binary format. If we also place the so-called *prefix property* condition on the encoding we use, which means that no such binary encoding may be be identical to the beginning of another such encoding, then it is relatively straight-forward to show that each such binary expression can be associated with a subinterval of [0,1], and that these subintervals do not overlap. The final step of deriving a well-formed probability invovles determining whether there are any subintervals of [0,1] which have not been put into correspondence with a lexicon, and dealing with the total length of such subintervals.

[8] See pseudocode in Figure 2.

most likely, a number of times in the string). In short, we batch process the text: we analyze the whole text several times [Line 2]. We begin with the "trivial" lexicon consisting of just the letters of the alphabet, but we build the lexicon up rapidly by iteration. On each iteration, we find the parse which maximizes the probability of the data, given the current hypothesized lexicon [Lines 3,4]. We consider as tentative new candidates for the lexicon a pair of "words" that occur next to each other in the string [Line 5]. We compute the description length of the entire string with and without the addition of the new lexical item, and we retain the candidate, and the new lexicon that it creates, if its retention leads to a reduction in the total description length [Line 8]. We set a reasonable stopping condition, such as having considered all adjacent pairs of words and finding none that satisfy the condition in Line 8.

1: Start condition: $\mathcal{L} \Leftarrow \Sigma$.
2: **repeat**
3: $\pi^* \Leftarrow \arg\max_{\pi \in \{parses\ of\ D\}} pr(\pi)$, given $\mathcal{L}$.
4: Assign a probability distribution over $\mathcal{L}$ based on the counts of words in π^*.
5: Choose at random two adjacent words, $w_{i_k} w_{i_{k+1}}$;
6: $w^* \Leftarrow w_{i_k} w_{i_{k+1}}$.
7: $\mathcal{L}^* \Leftarrow \mathcal{L} \cup w*$.
8: If DL(C,$\mathcal{L}^*$) < DL(C,$\mathcal{L}$), then $\mathcal{L} \Leftarrow \mathcal{L}^*$.
9: **until** Stopping Condition is satisfied.
10: **return** $\arg\max_{\pi \in \{parses\ of\ D\}} pr(\pi)$, given $\mathcal{L}$.

Figure 2

MDL-based approaches work quite well in practice, and as a selling point, they have the advantage that they offer a principled answer to the question of how and why natural language should be broken up into chunks. Many variants on the approach sketched here can be explored. For example, we could explore the advantages of a lexicon that has some internal structure, allowing words to be specified in the lexicon as concatenation of two or more other lexical entries; de Marcken's model permits this, thus encouraging the discovery of a lexicon whose entries are composed of something like morphs. We return to the general question shortly, in connection with the discovery of true linguistic morphology.

Hierarchical Bayesan models

In a series of recent papers, Goldwater, Johnson, and Griffiths (henceforth, GJG) have explored a different approach involving hierarchical Bayesian models, and they have applied this to the problem of inducing words, among other things (see Goldwater (2006), Goldwater *et al.* (2006), Johnson *et al.* (2006), Johnson (2008), and also Teh *et al.* (2006)). Like MDL models, these grammars are non-parametric models, which is to say, in the study of different

sets of data of the same *kind*, they consider models with different numbers of parameters—or to put it more crudely, the model complexity increases as we give the system more data. GJG describe the models by means of the process that generates them—where each model is a distribution, or a set of distributions over different bases—and what distinguishes this approach is that the history of choices made is cached, that is, made available to the process in a fashion that influences its behavior at a given moment. The process leans towards reproducing decisions that it has often made in the past, based upon a scalar *concentration* parameter $\alpha > 0$. After having already generated n words, the probability that we generate a *novel* word from an internal base word-generating distribution is $\frac{\alpha}{\alpha+n}$, a value that diminishes rapidly as the process continues. Conversely, the probability of generating word w_k which has already occurred $[w_k]$ times is $\frac{[w_k]}{n+[w_k]}$.

Such processes share with MDL models (though for quite different reasons) what sociologists call a *Matthew effect* (also called a *rich get richer* effect), whereby choices that have been selected over the past of the process are more likely to be selected in the future.

GJG use this process to model the generation of a lexicon, and Gibbs sampling to find the appropriate lexicon parameters.[9] We describe the simplest of their models, the unigram model, here. As we have noted, the process has three parameters: a concentration parameter α; a finite lexicon, i.e., a finite set of elements of Σ^*, each of which is associated with a parameter corresponding to how often that word has been seen in the data at this point; and a base distribution Φ over Σ^* used eventually to create new words for the lexicon.

We thus have two abstract objects to consider in connection with any long unbroken string S: one is a string b of 1's (yes) and 0's (no), as to whether a word-break occurs after the n^{th} symbol of S; and the other is the finite lexicon (which is what we are trying to figure out, after all). Given S and a particular b, a specific lexicon follows directly. The Gibbs sampling that is used provides a path from any initial set of assumptions about what b is (that is, any initial set of assumptions as to where the word breaks are) to essentially the same steady-state analysis of S into words. Here is how it does it, and it does not matter whether we begin with the assumption that there is a break between every symbol, between no symbols, or that breaks are initially assigned at random. We will iterate the following procedure until we reach equilibrium: we select an integer between 1 and $|S| - 1$, and calculate anew whether there *should* be a break there, i.e., we make a new decision as to whether b has a 0 or a 1 at position n, conditioned on the locations of the other breaks specified in b, which implicitly determines the words in the lexicon and their frequency. We do this by looking just at the righmost chunk to the left of position n and the leftmost chunk to the right of position n. For example, if the string is $...isawa - ca - t - inth - ewind...$ (where the hyphen indicates a break, and

[9] On Gibbs sampling, see Mackay (2002), for example.

no-hyphen indicates no break), and we choose to sample the position between *ca* and *t*, then we calculate the probability of two different strings *cum* breaks: the one just indicated, and this one: $...isawa - cat - inth - ewind....$ If we assume words are generated independently of their neighbors (the unigram assumption), then we need simply compare the probability of $ca - t$ and that of *cat*, and that decision will be made on the basis of the probability assigned by the process we have described. *That* probability, in turn, will not weigh heavily in favor of *cat* over *ca* and *t* early on, but if the corpus is in fact drawn from English, the counts of *cat* will begin to build up over those of *ca* and *t*, and the local decision made at position *n* will reflect the counts for all of the words that occur, given the analysis so far, in the string. GJG show that if we drop the unigram assumption about words, and assume essentially that the probability of each word is conditioned by the preceding word (more accurately, that the parameters of the Dirichlet process selecting a word are conditioned by the preceding word) and if we let the distribution Φ that proposes new words itself adapt to the language's phonotactics (which can be learned from the lexicon), then results are considerably improved.

2.3 Word Boundary detectors

The very first work on explicit development of boundary detectors was due to Zellig Harris, but his primary application of the notion was to morpheme-detection, which we will return to shortly. Nonetheless, his ideas have inspired many subsequent workers, who have looked to see if there were local characteristics, detectable within a small window of 5 or 6 letters, which would give a strong indication of where a word boundary falls in a text. We will look at one recent example, that of Cohen *et al.* (2002), which makes an effort to detect word boundaries both by finding likely boundary points and by finding likely word sequences. Cohen *et al.* (see also Cohen *et al.* (2007)) let their hybrid model vote on the best spot to hypothesize a break to be, and so they call this a Voting Expert model. Their expert uses the log frequency of a conjectured word-chunk *w* as its measure of goodness as a word, and their measure of whether a point *i* is a good break-point is what they call the boundary entropy, defined roughly (but only roughly) as the entropy of the frequency distribution of individual letters that follow the hypothesized word that ends at point *i*. Thus, if a string $thusifastringisanalyzedat...$ is analyzed at point 13 as containing the word *string* stretching from point 7 to point 13, then we compute the frequency of all of the letters that in fact follow the string *string* somewhere in the corpus. The greater the entropy is of that multiset, the likelier it is that the ending point of *string* is a word boundary (on the grounds that words are relatively poor as regards their ability to impose a decision on what letter should come next). This is the entropic version, employed by Hafer and Weiss 1974, of Harris's successor frequency notion, to which we return in the next section. Cohen *et al.* take into account phonological frequency effects by not using observed frequencies, but rather corresponding

z-scores. For any sequence s of letters of length $|s|$ and frequency $f(s)$, they know both the average frequency $\mu(|s|)$ of distinct observed strings of length $|s|$ in the text, and the standard deviation from this mean of all of the strings of length $|s|$, so everywhere where one would expect to put a probability, they use a z-score instead (i.e., $\frac{freq(s)-\mu(|s|)}{\sigma}$): the measure of goodness of a chunk is the logarithm of that value, and the familiar notion of conditional entropy is modified to use this sort of z-score instead of a frequency.

The algorithm makes one pass over the string, shifting a window of limited size (the authors give an example of width 3, but in actual applications they use a window of 6, or more); at each stop of the window, the two measures (goodness of chunk, goodness of break point) each independently select the point within the window which maximizes their own measure, but looking only at the small substring within the window (and not, for example, what had been decided earlier on in the pass to segments "to the left," so to speak, except insofar as that information is implicit in the accrued voting). When the string has been scanned, a (non-linear) counting process decides how the votes which have been assigned to each point between the letters by the two measures should be transformed into hypotheses regarding word breaks.

2.4 Successes and failures in word segmentation

The largest part of the failures of all approaches to word segmentation are failures of level rather than failures of displacement: that is, failures are typically either of finding chunks that are too *large*, consisting of common pairs of words (*ofthe*, *NewYork*) or of not so common words composed of common pieces (*commit ment*, *artificial ly*), rather than errors like *c hunks*, though those certainly do appear as well. The most interesting result of all of the work in this area is this: there is no way to solve the word segmentation problem without also making major progress with the problem of automatic learning of morphology and syntax. Knowledge of the statistical properties of strings can be used to infer words only to the extent that the device that generated the strings in the first place *used* knowledge of words, and only knowledge of words, to generate the string in the first place; and in actual fact, the systems that generate our natural language strings employ systems at several levels: it is not words, but *morphemes* that consist of relatively arbitrary sequences of letters, and words are the result of a system responsible for the linear placement of morphemes. In addition, there is a system responsible for the sequential placement of words—we call it *syntax*—and it too has a great impact on the statistics of letter placement. A system that tries to learn the structure of language on the basis of a model that is far poorer than the real structure of language will necessarily fail—we may be impressed by how well it does at first, but failure is inevitable, unless and until we endow the learning algorithm with the freedom of thought to consider models that take into consideration the structure that indeed lies behind and within language. In the next section, we turn to the task of learning morphological structure.

3 Unsupervised learning of morphology.

In this section, we will discuss the automatic learning of morphology. Most of the attention in this part of the literature has gone to the problem of segmentation, which is to say, the identification of the morphs and morphemes of a language, based entirely on naturalistic corpora. The identification and treatment of morphosyntactic features is an additional problem, which we shall touch on only in passing (though it is a real and important challenge). When we look at a sample of English, we certainly want to discover that *jumps* and *jumping* consist of a stem *jump* followed by the two suffixes *s* and *ing*, and a solution to the problem of identifying morphs gives us that information; but at the same time, we would like to know that $-s$ marks the `3rd person singular present` form of the verb. Such information goes well beyond the problem of segmentation, and brings us to the domain of morphosyntactic information, an area in which relatively little has been done in the domain of unsupervised learning of morphology.

3.1 Zellig Harris

All discussions of the problem of automatic segmentation aiming at discovering linguistically relevant units start with the work of Zellig Harris. In the mid 1950s, he noticed that one could define a function that, informally speaking, specifies how many alternative symbols may appear at any point in the string, given what has preceded. In light of both the method and the date, it is impossible not to sense an inspiration from Shannon's work, which had just appeared (Shannon & Weaver (1949)). Harris himself published two papers addressing this approach (1955, 1967), and returned to it briefly in other works till the end of his life Harris (1991). At least as interesting for our purposes was the computational implementation of Harris's idea in Hafer & Weiss (1974). The presentation in this chapter relies primarily on Harris 1967 and Hafer and Weiss. We consider a family of Harrisian algorithms for segmenting words into morphemes, given a sample of words $\mathcal{W} = \{w_i\}$ from a language, where each $w_i \in \Sigma^*$, for some alphabet Σ. We wish to associate a real value with the position that lies between each symbol in each word, and while we can imagine several slightly different ways to do this, the ways all attempt to capture the idea of measuring how many different ways the string to the left can be continued, in view of what we know about the entire set of words $\mathcal{W}$. The core notion of *successorfrequency* is defined as follows: The successor frequency $\mathrm{SF}(p, \mathcal{W})$ of a string p in a set of words $\mathcal{W}$ is 0 if no words in $\mathcal{W}$ begin with p (i.e., there is no w in $\mathcal{W}$ which can be expressed as $p\alpha$ where $\alpha \in \Sigma^*$), and, more interestingly, it is equal to the number of distinct symbols $\{l_1, ..., l_k\}$ all of which can follow the prefix p in $\mathcal{W}$: that is, our successor frequency is the size of the set $\{l_1, ..., l_k\}$ such that pl_i is a prefix of a word in $\mathcal{W}$. A similar definition can be constructed to define the predecessor frequency in

a mirror-image fashion, specifying how many different letters can immediately precede any given word-final substring.

Harris's intuition was that the successor frequency was high, or relatively high, at points in a string corresponding to morpheme boundaries, and the same is true for the predecessor frequency. But if the method works well in many cases, it fails in many others as well, either due to data sparsity, or to other effects. One such effect arises when the set of suffixes to a given stem all sharing a common letter (for example, the words *construction* and *constructive* have a peak of successor frequency after *constructi*, and a peak of predecessor frequency after *construct*).

Hafer & Weiss (1974) tested 15 different interpretations of Harris's algorithm, and found a wide variety in precision and recall of the interpretations, ranging from "completely unacceptable" when cuts were made at thresholds of successor frequency, to as high as 91% precision and recall of 61%; this was the result of making a morpheme cut when either of two conditions was met: (a) the word up to that point was also a free standing word, and the predecessor frequency there was 5 or greater; or (b) the successor frequency at the point was greater than 1, and the predecessor frequency was greater than 16. One can see that some effort was expended to tune the parameters to suit the data.

3.2 Using description length

Anybody's list of words in a language, no matter how it is obtained, contains a great deal of redundancy, for all of the reasons that we discussed in the first section of this paper: morphological roots appear with a variety of prefixes and suffixes, but that variety is limited to a relatively small number of patterns. The discovery of the morphological structure of a language is essentially the discovery of this kind of redundancy in the lexicon; removing the redundancy will both shorten the description of the lexicon and take us closer to an accurate characterization of the morphology. For example, if a language (in this case, English) contains a set of words *walk, walks, walked, walking, jump, jumps, jumped, jumping*, then rather than expressing all eight of the words separately, we can achieve greater simplicity by extracting the redundancy inherent in the data by identifying two stems, *walk* and *jump*, and four suffixes, $\emptyset$, *s*, *ed*, *ing*. See Figure 3.

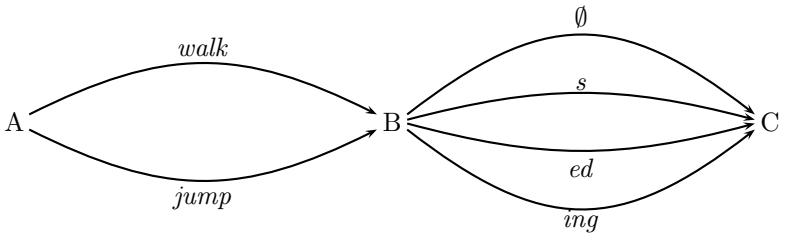


Figure 3

But how do we turn that intuition into a computationally implemented approach? We employ a hill-climbing strategy, and focus on deciding what determines the shape of the landscape that we explore, and how to avoid discovering local maxima that are not global maxima.

We specify first what the formal nature is of our morphology—typically, a finite state automaton (for a definition of an FSA, see Chapter One[?], and section 4.1 below). Given the nature of a morphology, it is always *possible* to treat a word as an unanalyzed morpheme, and an algorithm will generally begin by assuming that in the initial state, the morphology treats all words of the corpus C as unanalyzable single morphemes (Step 1) (see Figure 4). We will call that state of a morphology $\mathcal{M}_0(C)$. Some approaches take a one-time initial step of analysis (Step 2), with the aim of avoiding local optima that are not global optima. A loop is entered (Steps 3-5) during which hypotheses about morphological structure are entertained; any such hypothesis may either be dependent on what the current hypothesized morphology $\mathcal{M}$ is, or simply be based on information that is independent of the current $\mathcal{M}$ by using information that was already available in $\mathcal{M}_0$, such as how often a string occurs in the corpus C (Step 4). A simplicity-based approach defines some notion of simplicity—the function $C(\mathcal{M}, \mathcal{M}')$ in line 5—to decide whether the modified morphology is preferable to the current hypothesis.

1: Start condition: $\mathcal{M} \Leftarrow \mathcal{M}_0(C)$.
2: Some approaches: bootstrap operation $\mathcal{B}$: $\mathcal{M} \Leftarrow \mathcal{B}(\mathcal{M}_0(C))$.
3: **repeat**
4: $\quad \mathcal{M}' = \mathcal{F}(\mathcal{M}, \mathcal{M}_0)$
5: $\quad$ If condition $C(\mathcal{M}, \mathcal{M}')$ is satisfied, $\mathcal{M} \Leftarrow \mathcal{M}'$.
6: **until** Stopping Condition

Figure 4

A method that employs Minimum Description Length analysis,[10] such as Goldsmith (2001) and Goldsmith (2006), will use description length of the data as the condition in Step 5: a new morphology is preferred to the current one if the description length of the data is decreased with the new morphology, where the description length of the data is the sum of two quantities: (i) the inverse log probability of the data, and (ii) the length of the morphology as measured in bits. How are these quantities calculated?

In order to calculate the probability of a word that the FST generates, we must associate a probability distribution with each set of edges that leave a given state. Because a word, considered as a string of letters, typically is associated to only one path through the FST, it is easy to assign probabilities based on observed frequencies; we count the number of times each edge (n_i, n_j) is traversed when all of the data is traversed, and then assign the probability of going to node n_j, given that we are at node n_i, as $\frac{Count(n_i,n_j)}{\sum_k Count(n_i,n_k)}$, and the probability of any path through the FST (hence, of any word that it generates) is the product of the probabilities associated with each edge on the path. The number of bits required to encode such a word is, then, the negative log probability of the word, as just calculated.

How should we compute the length of a morphology in bits? In the discussion of MDL above concerning word segmentation, we saw one natural way to compute the length of a list of strings, and nothing would be simpler than to count the number of bits that were required in order to spell out the list of labels on each of the edges. The result would be $log_2 N + \sum_{i=1}^{N} |w_i| \; log_2 \; m$, where N is the total number of morphemes, m the number of symbols in the alphabet, and the set of labels is the set $\{w_i\}$.

In our discussion of MDL in connection with word discovery, we suggested that discovering the right lexicon is a matter of finding the right balance between a lexicon that was not over-burdened and an encoding of the data that was not too intricate. Now we are saying something more: we are recognizing that we were not ambitious enough when we took it for granted that a lexicon was a list of words, because any lexicon that simply lists all its members is highly redundant in much the same way that a text in a particular language that is not analyzed into words is highly redundant. We therefore turn to the discovery of morphology as the means to reduce the redundancy in the lexicon.

This suggests a broader understanding of the cost of designing a particular morphology for a set of data: use information theory in order to make explicit what the cost of every single piece of the morphology is—which is to say, the morphology is a labeled, probabilistic FSA. It consists of a set of states $\{N\}$ (including a start state, and a set of final, or accepting, states), a set of edges E $\subset N \times N$, a list of morphemes, and a label on each edge consisting of a

[10] An open source project that implements such an approach can be found at `http://linguistica.uchicago.edu`.

pointer to an item on the morpheme list. The cost of the list of morphemes is calculated just as we discussed in section 2.2, in the context of a lexicon of words. More interesting is the complexity, or cost, in bits associated with the FSA itself, which is the sum of the cost of each edge (n_i, n_j) and its associated label l_m. The cost of the edge is $-(log\, pr_n(n_i) + log\, pr_n(n_j))$, where $pr_n()$ is a probability distribution over nodes, or states, based on how often each node is traversed in parsing the data, as discussed four paragraphs above. The cost in bits of the pointer to an item on the morpheme list is equal to $-log\, pr_\mu(m)$, where $pr_\mu()$ is a probability distribution over the items in the morpheme list, based on how many words in the corpus are generated by a path which includes an edge pointing to morpheme m.[11]

3.3 Work in the field

There has been considerable work in the area of automatic learning of morphology since the 1950s, and quite a bit of it in the last fifteen years or so. Subsequent to Zellig Harris's work mentioned earlier, there was work by Nikolaj Andreev (1965 and 1967, described in Cromm (1997)) in the 1960s, and later by de Kock and Bossaert 1969, 1974. An interesting paper by Radhakrishnan (1978) in the text compression literature foreshadows some of the work that was yet to come. Several years later there was a series of paper by Klenk and others (1989, also Wothke & Schmidt (1992) and Flenner (1995)) which focused on discovery of local segment-based sequences which would be telltale indicators of morpheme boundaries. More recently, there has been a series of papers by Medina Urrea (2000, 2005, 2006), and appproaches employing MDL have been discussed in van den Bosch *et al.* (1996), Kit & Wilks (1999), Goldsmith (2001, 2006), Baroni 2003, and Argamon et al. 2004. Important contributions have been made by Yarowsky and Wicentowski (see Yarowsky & Wicentowski (2000)), by Schone and Jurafsky 2001, and Creuz and colleagues (see 2002, 2003, 2004, 2005a, and 2005b).

Clark (2001a, 2001b, 2002) explored the use of stochastic finite-state automata in learning both the concatenative suffixal morphology of English and Arabic, but also the more challenging case of strong verbs in English and broken plurals in Arabic, using expectation-maximization [cross-reference in this book] to find an optimal account, given the training data.[12] It is interesting to note that this work takes advantage of the active work in bioinformatics based on extensions of hidden Markov models, which itself came largely from the speech recognition community. Memory-based approaches such as

[11] That is, if there are a total of V words in the corpus, and a total of M morphemes in all of the M words (V, but not M, depends on the morphology that we assume), and if $K(m)$ is the number of words that contain the morpheme m, then the cost of a pointer to that morpheme is equal to $pr_\mu(m) = log_2 \frac{M}{K(m)}$.

[12] A number of studies have dealt with Arabic, such as Klenk (1994); see also van den Bosch et al. (2007) and other chapters in that book.

van den Bosch & Daelemans (1999) (on Dutch) employ rich training data to infer morphological generalizations extending well past the training data.

Algorithms that learn morphology in a strictly unsupervised way are never certain about what pairs of words really are morphologically related; they can only make educated guesses, based in part on generalizations that they observe in the data. Some researchers have explored what morphological learners might be able to do if they were told what pairs of words were morphologically related, and the systems would have to induce the structure or principles by which they were related. This work is not strictly speaking unsupervised learning; the learner is helped along considerably by knowledge that pairs of words are indeed morphologically related.

Let us suppose, for purposes of discussion, that we have determined in some fashion that the pairs of words that we are given are largely distinguished by material on the right-hand side of the word: that is, that the system is largely suffixal. Then, given any pair of related words w_1 and w_2, where w_1 is not longer than w_2, then there are at most $|w_1+1|$ ways to account for the pairing, based solely on treating the relationship as based on morphs. Given the pair $jump/jumped$, there are five generalization we might consider, each specifying a pair of suffixes in that pair of words: $\emptyset/ed, p/ped, mp/mped, ump/umped$, and $jump/jumped$. As we consider a large set of pairs of words, it is not hard to see that the correct generalization will generally be the one which occurs most frequently among a large number of word-pairs. This approach has the advantage that it can say something useful even about generalizations that involve a very small number of pairs (e.g., $say/said, pay/paid$); this is more difficult for a purely unsupervised approach, because it is difficult for a purely unsupervised approach to become aware that those pairs of words *should* be related, so to speak. An early effort along these lines was Zhang & Kim (1990). Research employing inductive logic programming to deal with this problem by automatically creating decisions lists or trees has included Mooney & Califf (1996) (on English strong verbs), Manandhar *et al.* (1998), Kazakov & Manandhar (1998) (an approach that also employs an algorithmic preference for simpler morphological analyses), Kazakov (2000)—which presents a very useful survey of work done in the 1990s on computational morphology—Erjavec & Džeroski (2004), which discusses the case of Slovene in some detail, and Shalonova & Flach (2007) (English and Russian). Baroni et al. (2002) took an interesting step of using mutual information between pairs of nearby words in a corpus as a crude measure of semantic relatedness. It had been noticed (Brown *et al.* (1992)) that words that are semantically related have a higher probability than chance to occur within a window of 3 to 500 words of each other in a running text, and they explored the consequences for analyzing pairs of words (looking at English, and at German) that are both formally similar and with relatively large point-wise mutual information. This work therefore looks a lot like the work described in the preceding paragraph, but it does so in a rigorously unsupervised way.

4 Implementing computational morphologies

Sophisticated computational accounts of natural language morphology go back more than forty years in the literature; we can still profitably read early articles such as that by P. H. Matthews (1966).

There have been several excellent book-length studies of computational morphology in recent years, with considerable concern for actual, real-world implementation, notably by Beesley & Karttunen (2003) and Roark & Sproat (2006), as well as Ritchie *et al.* (1992) and Sproat (1992). Current work in this area focuses to a very large extent on the use of finite-state transducers as a means to carry out the functions of morphology. This work was stimulated by the work of Douglas Johnson 1972, Ronald Kaplan and Martin Kay 1981, Kimmo Koskeniemmi 1983, and developed in a number of places by Lauri Karttunen and colleagues 1993. Beesley and Karttunen's recent book is especially detailed and lucid, and contains Xerox software that can be used by the reader.

A well functioning computational morphology for a language can be vital for many practical applications. Spell-checking is a humble but honest function of many products appreciated by a wide range of end-users, and in a language with a rich inflectional morphology, as we find in languages such as Finnish, Hungarian, Turkish, the Bantu languages, and many others, the total number of possible forms that a user might reasonably generate is far greater than the capacity of a computer to hold in its memory, unless the entire family of forms is compressed to a manageable size by virtue of the redundancies inherent in a computational morphology. It is typically the inflectional morphology which gives rise to the very large number of possible forms for nouns and verbs, and it is typically inflectional morphology which can most usefully be stripped off when one wishes to build a document-retrieval system based not on actual words, but on the most useful part of the words. Syntactic parsing in most languages requires a knowledge of the morphosyntactic features carried by each word, and that knowledge is generally understood as being wrapped up in the morphology (primarily the inflectional morphology) of the language.[13] A number of researchers have explored the effect on the quality of information and document retrieval that is produced by incorporating knowledge of inflectional and derivational morphology, including Harman (1991),

[13] Even for a language like English, in which the morphology is relatively simple, and one could in principle not do *too* badly in an effort to list all of the inflected forms of the known words of English, the fact remains that it is rarely feasible in practice to construct a list of all such words simply by data-scraping—that is, by finding the words in nature. To be sure that one had obtained all of the inflected forms of each stem, one would have to build a morphology to generate the forms, thereby bringing us back to the problem of building a morphology for the language.

Krovetz (2000), Hull (1996), Kraaij & Pohlmann (1996), Xu & Croft (1998), Goldsmith *et al.* (2001), Larkey (2002), and Savoy (2006).

Let us consider briefly how a practical system can be overwhelmed by the size of natural language word sets if morphology is not addressed in a systematic way. In Swahili, a typical Bantu language in this regard, a verb is composed of a sequence of morphemes. Without pretending to be exhaustive, we would include an optional prefix marking negation, a subject marker, a tense marker, an optional object marker, a verb root, a choice of zero of more derivational suffixes marking such functions as causative, benefactive, and reflexive, and ended by a vowel marking mood. Subject and object markers are chosen from a set of approximately 20 options, and tenses from a set of about 12. Thus each verb root is a part of perhaps 100,000 verbs. Similar considerations hold in many languages—in fact, almost certainly in the great majority of the world's languages.

4.1 Finite state transducers

A large part of the work on computational morphology has involved the use of finite-state devices, including the development of computational tools and infrastructure. Finite state methods have been used to handle both the strictly morphological and morphotactic, on the one hand, and the morphophonology and graphotactics on the other. We have already encountered the way in which strictly morphological information can be implemented with a finite state automaton, as in Figure 3. By extending the notion of finite state automaton to that of finite state *transducer*, we can use much the same notions in order to not only generate the correct surface morphemes; we can also create a device that can map surface sequences of letters (or phones) to abstract morphosyntactic features such as NUMBER and TENSE.

Computational morphology has also applied the notion of finite state transducer (the precise details of which we return to shortly) to deal with the problem of accounting for regularities of various sorts concerning alternative ways of realizing morphemes. For example, both the English nominal suffix marking PLURAL and the English verbal suffix marking 3RD PERSON SINGULAR is normally realized as *s*, but both are regularly realized as *es* after a range of stems which end in *s, sh, ch,* and *z*.

We refer to these two aspects of the problem as *morphotactics* and *phonology*, respectively. Two methods have been developed in considerable detail for the implementation of these two aspects within the context of finite state devices. One, often called "two level morphology," is based on an architecture in which a set of constraints are expressed as finite-state transducers that apply in parallel to an underlying and a surface representation. Informally speaking, each such transducer acts like a constraint on possible differences that are permitted between the underlying and the surface labels, and as such, any paired underlying/surface string must satisfy all transducers. The other approach involves not a *parallel* set of finite-state transducers, but rather a

cascaded set of finite-state transducers, which can be compiled into a single transducer. A lucid history of this work, with an account of the relationship between these approaches, can be found in Karttunen & Beesley (2005); a more technical, but accessible, account is given by Karttunen (1993).

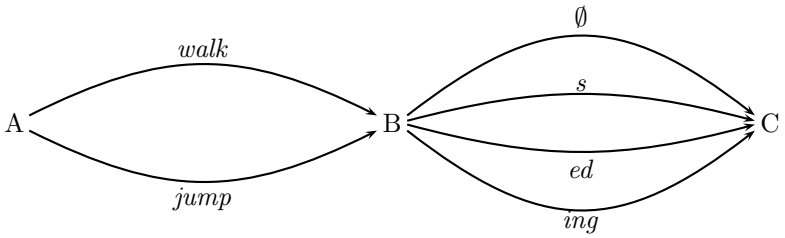
The term "two-level morphology," due to Koskenniemi (1983), deserves some explanation: by its very name, it suggests that it is possible to deal with the complexities of natural language morphology (including morphophonology) without recourse to derivations or intermediate levels. That is, formal accounts which are influenced by generative linguistics have tended uniformly to analyze language by breaking up phenomena into pieces that could be thought of as applying successively to generate an output from an input with several intermediate stages. It would take us too far afield to go through an example in detail, but one could well imagine that the formation of the plural form of *shelf* could be broken up into successive stages: $shelf \rightarrow shelf + s \rightarrow shelv + s \rightarrow shelves$. Here, we see the suffixation of the plural '*s*' happening (in some sense!) first, followed by the change of f to v, followed in turn by the insertion of e. In contrast, finite-state automata offer a way of dealing with the central phenomena of morphology without recourse to such a step-by-step derivation: hence the term 'two-level morphology,' which employs only two levels: one in which morphosyntactic features and lexical roots are specified, and one which matches the spelled (or pronounced) form of the word. We return to this in Section 4.2.

The notion of *finite state automaton* (often abbreviated as *FSA*) was first presented in Kleene (1956), itself inspired by the work of McCulloch & Pitts (1943) some ten years earlier. An FSA is a kind of directed graph: a directed graph is by definition a finite set of nodes $\mathcal{N}$, along with a set of edges E, where an edge is an ordered pair of nodes. *Nodes* in an FSA are often called *states*. For a directed graph to be an FSA, it must be endowed with three additional properties: it must have a distinguished node identified as its *start* state; it must have a set of one or more *stopping* (or *accepting*) states; and it must have a set of labels, $\mathcal{L}$, with each edge associated with exactly one label in $\mathcal{L}$. While $\mathcal{L}$ cannot in general be null, it may contain the null string as one of its members. In purely mathematical contexts, it is convenient to assume that each label is an atomic element, indivisible, but in the context of computational linguistics, we rather think of $\mathcal{L}$ as a subset of Σ^*, for some appropriately chosen Σ. In that way, the morphs of a given language (e.g., *jump, dog, ing*) will be members of $\mathcal{L}$, as will be descriptions of grammatical feature specifications, such as `1st person` or `past-tense`.

When we explore an FSA, we are typically interested in the set of paths through the graph, and the strings associated with each such path—we say that a path *generates* the string. A path in a given FSA is defined as a sequence of nodes selected from $\mathcal{N}$, in which the first node in the sequence is the starting state of the FSA, the last node in the sequence is one of the stopping states of the FSA, and each pair of successive nodes (n_i, n_{i+1}) in the sequence corresponds to an edge e_j of the FSA. We associate a string S with a

path p simply by concatenating all of the labels of the edges corresponding to the successive pairs of nodes comprising p. If we take a grammar of a language to be a formal device which identifies a set of grammatical strings of symbols, then an FSA is a grammar, because it can be used to identify the set of strings that correspond to all paths through it. Given a string S in $\mathcal{L}^*$, we can identify all paths through the FSA that generate S.

Finite state morphologies employ a generalization of the finite-state automaton called a finite state *transducer*, or *FST*, following work by Johnson (1972). An FST differs from an FSA in that an FST has two sets of labels (or in principle even more, though we restrict the discussion here to the more common case), one called *underlying* labels, $\mathcal{L}_\mathcal{U}$, and one called *surface* labels, $\mathcal{L}_\mathcal{S}$, and each edge is associated with a *pair* of labels (l_U, l_S), the first chosen from the underlying labels, and the second from the surface labels—it is traditional, however, to mark the pair not with parenthesis, but with a simple colon between the two: $l_U : l_S$. The FSA thus serves as a sort of translation system between $\mathcal{L}^*_\mathcal{U}$ and $\mathcal{L}^*_\mathcal{S}$. In fact, an FST can be thought of as *two* (or even more) FSAs which share the same nodes, edges, starting states and stopping states, but which differ with regard to the labels associated with each edge, and we only care about looking at pairs of identical paths through these two FSAs. The beauty of the notion of FST lies in the fact that it allows us to think about pairs of parallel paths through otherwise identical FSAs as if they were just a single path through a single directed graph. For this reason, we can say that FSA are *bidirectional*, in the sense that they have no preference for the underlying labels or the surface labels: the same FST can translate a string from $\mathcal{L}^*_\mathcal{S}$ to $\mathcal{L}^*_\mathcal{U}$, and also from $\mathcal{L}^*_\mathcal{U}$ to $\mathcal{L}^*_\mathcal{S}$. If we construct an FST whose second set of labels is not underlying forms but rather category lables, then the same formalism gives us a parser: tracing the path of a string through the FST associates the string with a sequence of categories. Finite state automata are relatively simple to implement, and very rapid in their functioning once implemented. See Figure 5.



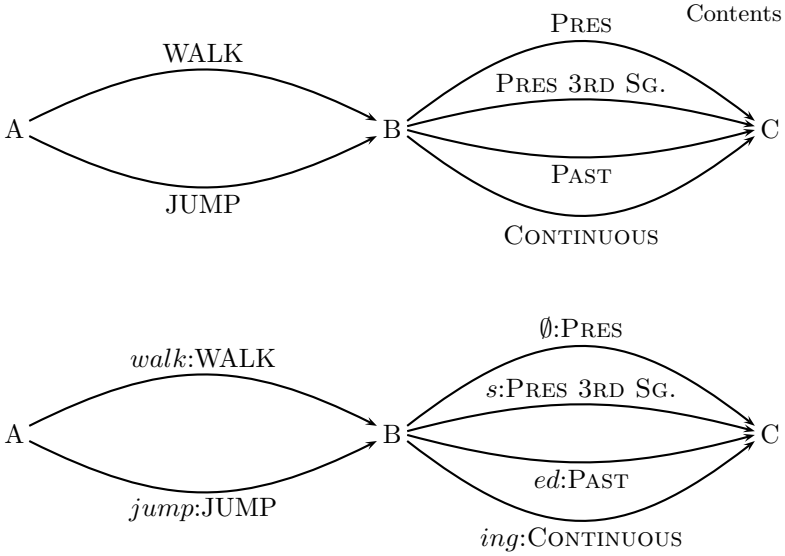


Figure 5

4.2 Morphophonology

Rare is the language which does not contain rules of its spelling system whose effect is to vary the spelling of a morpheme depending on the characteristics of the neighboring morphemes. English has many such cases: as we noted, some English words, like *wife* and *knife*, change their f to v in the plural; the plural suffix itself is sometimes s and sometimes es, depending on what precedes it. Since these patterns tend to recur within a given language, it is traditional to analyze this by saying that there is a single underlying label for the morpheme, but two or more surface labels that the transducer relates to the single underlying label.

Koskenniemi (1983) developed a system of notation widely used in finite state morphophonology to deal with this. The challenge of the task is to make explicit when any departure from simple identity is required, or permitted, between the underlying label u and the surface label s. Koskenniemi's idea was that for any given pairing like $f : v$, we can define a *context* that either permits or requires that correspondence, where by a *context* we mean a specification of the symbols that appear to the left and to the right, on both the underlying labels and the surface labels. For example, if every occurrence of underlying t corresponds to a surface s when and only when an i follows on *both* the underlying and surface labels, then we can specify this thusly: $t : s \Leftrightarrow \_\_ i : i$. If we wanted to express the generalization that when two vowels were adjacent on the string of underlying labels, only the second of them appears among the surface labels, then we would represent it this way: $V : \emptyset \Leftarrow \_ V :$, where V is a cover symbol standing for any vowel. The "$\Leftarrow$" is taken to mean that in the

context described to the right of the arrow, any occurrence of the underlying label in the pair on the left *must* be realized as the surface label of the pair on the left (in this case, as the null symbol). If the arrow pointed to the right, as in $V : \emptyset \Rightarrow \_\_V$:, the rule would be saying that the correspondence of underlying V to surface $\emptyset$ can only occur when an underlying vowel follows.

In the case of *wife/wives*, we must account for the pair *(f:v)*. Since this same pairing is found in a good number of English words, an appealing way to formalize this is to specify that the morpheme underlying *wife* contains a special symbol, which we indicate with a capital F: *wiFe*. An underlying F corresponds to a surface v, when the plural suffix follows, or in all other cases to a surface *f*. If the underlying form of the plural form of *wife* is **wiFe+NounPlural**, then we can express this as: $F : v \Leftarrow e + NounPlural$:, and the associated surface label will be *wives*.

5 Conclusions

The computational study of morphology is of interest because of its importance in practical applications, both present and future, and because of its theoretical interest. We have seen that the words of a language are not simply a fixed set of strings chosen from a language's inventory of letters, or phonemes; the vocabulary is in fact built out of a set of morphemes in ways that are potentially quite complex, and in ways that may give rise to complex modifications of one or more morphemes in a word, each modification of each morpheme potentially sensitive to the choices of each of the other morphemes in the word.

While the number of distinct words in a language does not grow as rapidly with length as the number of sentences in a language does, it is nonetheless true that the size of the total lexicon of a language is vastly larger than the size of the set of morphemes used to generate those words. In order to ensure that a system handles the entire lexicon, it is both practically and theoretically necessary to generate the lexicon computationally in a way that reflects the true structure of the morphology of the language. In addition, the meaning and function of a word is in many respects decomposable into the meaning and function of its inflectional stem and inflectional affixes, and so morphological analysis is an important step in statistical and data-driven methods of machine translation, at least in languages with rich morphologies.

At the same time, the formal structure of the morphological grammar of a language may be quite a bit simpler than the syntactic grammar, and allow for greater success at this point in the task of automatically inferring the morphology from data with relatively little hand-tagging of the data or contribution on the part of a human linguist. At present, the work on unsupervised learning in this area has focused on the problem of segmentation, but work is certain to procede in the direction of choosing the correct structure among alternative candidate FSAs, given a training corpus. Advances in this area will shed light on the more general problem of induction of regular languages, which in turn may be helpful in the goal of induction of more comprehensive grammars from natural language corpora.

References

Ando, Rie Kubota & Lillian Lee (2003), Mostly-unsupervised statistical segmentation of Japanese kanji sequences, *Journal of Natural Language Engineering* 9:127–149.

Andreev, Nikolaj D. (ed.) (1965), *Statistiko-kombinatornoe modelirovanie jazykov*, Moskow/Leningrad.

Andreev, Nikolaj D. (1967), *Statistiko-kombinatorney metody v teoretičeskom i prikladnom jazykovedenii*, Leningrad.

Argamon, Shlomo, Navot Akiva, Amihood Amir, & Oren Kapah (2004), Efficient unsupervised recursive word segmentation using minimum description length, in *In Proc. 20th International Conference on Computational Linguistics (Coling-04*, (22–29).

Baroni, M., J. Matiasek, & H. Trost (2002), Unsupervised discovery of morphologically related words based on orthogrpahic and semantic similarity, in *Proceedings of the Workshop on Morphological and Phonological Learning*, SIGPHON-ACL, (11–20).

Baroni, Marco (2003), *Distribution-driven morpheme discovery: a computational/experimental study.*

Beesley, Kenneth R. & Lauri Karttunen (2003), *Finite-State Morphology: Xerox Tools and Techniques*, CSLI.

Brent, Michael (1999), An efficient, probabilistically sound algorithm for segmentation and word discovery, *Machine Learning* 34(1-3):71–105.

Brent, Michael R. & Timothy A. Cartwright (1996), Distributional regularity and phonotactics are useful for segmentation, *Cognition* 61:93–125.

Brent, Michael R., Sreerama K. Murthy, & Andrew Lundberg (1995), Discovering morphemic suffixes : A case study in MDL induction, in *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, (find pages).

Brown, Peter F., Peter V. Desouza, Robert L. Mercer, & Jenifer C. Lai (1992), Class-based n-gram models of natural language, *Computational Linguistics* 18:467–479.

Cairns, P., R. Shilcock, N. Chater, & J. Levy (1997), Bootstrapping word boundaries: a bottom-up corpus-based approach to speech segmentation, *Cognitive Psychology* 33:111–153.

Cartwright, Timothy & Michael Brent (1994), Segmenting speech without a lexicon: The roles of phonotactics and the speech source, in *Proceedings of the First Meeting of the ACL Special Interest Group in Computational Phonology*, SIGPHON-Association for Computational Linguistics, Las Cruces, (83–91).

Clark, Alexander (2001a), Learning morphology with pair hidden markov models, in *ACL (Companion Volume)*, (55–60).

Clark, Alexander (2001b), Partially supervised learning of morphology with stochastic transducers, in *NLPRS*, (341–348).

Clark, Alexander (2002), Memory-based learning of morphology with stochastic transducers, in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL*, (513–520).

Cohen, Paul, Niall Adams, & Brent Heeringa (2007), Voting experts: An unsupervised algorithm for segmenting sequences, *Intell. Data Anal.* 11(6):607–625, ISSN 1088-467X.

Cohen, Paul R., Brent Heeringa, & Niall M. Adams (2002), An unsupervised algorithm for segmenting categorical timeseries into episodes, in *Proceedings of the ESF Exploratory Workshop on Pattern Detection and Discovery*, Springer-Verlag, London, UK, ISBN 3-540-44148-4, (49–62).

Creutz, Mathias (2003), Unsupervised segmentation of words using prior distributions of morph length and frequency, in Erhard Hinrichs & Dan Roth (eds.), *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, (280–287).

Creutz, Mathias & Krista Lagus (2002), Unsupervised discovery of morphemes, in *Proceedings of the Workshop on Morphological and Phonological Learning of ACL-02*, SIGPHON-ACL, Philadelphia, (21–30).

Creutz, Mathias & Krista Lagus (2004), Induction of a simple morphology for highly-inflecting languages, in *In Proceedings of 7th Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON*, (43–51).

Creutz, Mathias & Krista Lagus (2005a), Inducing the morphological lexicon of a natural language from unannotated text, in *In Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR05*, (106–113).

Creutz, Mathias & Krista Lagus (2005b), Unsupervised morpheme segmentation and morphology induction from text corpora using morfessor 1.0, in *Helsinki University of Technology*, (2005).

Cromm, Oliver (1997), Affixerkennung in deutschen wortformen, *LDV Forum* 14(2):4–13.

Ellison, T. Mark (1994), *The Machine Learning of Phonological Structure*, Ph.D. thesis, University of Western Australia.

Erjavec, Tomaž & Sašo Džeroski (2004), Machine learning of morphosyntactic structure: lemmatizing unknown Slovene words, *Applied Artificial Intelligence* 18:17–40.

Flenner, Gudrun (1995), Quantitative morphsegmentierung im spanischen auf phonologischer basis, *Sprache und Datenverarbeitung* 19(2):3–79.

Goldsmith, John (2001), Unsupervised learning of the morphology of a natural language, *Computational Linguistics* 27(2):153–198.

Goldsmith, John A. (2006), An algorithm for the unsupervised learning of morphology, *Natural Language Engineering* 12(4):353–371.

Goldsmith, John A., Derrick Higgins, & Svetlana Soglasnova (2001), Automatic language-specific stemming in information retrieval, in *In Cross-language information retrieval and evaluation: Proceedings of the CLEF 2000 workshop*, Springer Verlag, (273–283).

Goldsmith, John A. & Jeremy O'Brien (2006), Learning inflectional classes, *Language Learning and Development* 2(4):219–250.

Goldwater, Sharon (2006), *Nonparametric Bayesian Models of Lexical Acquisition*, Ph.D. thesis, Brown University.

Goldwater, Sharon, Thomas L. Griffiths, & Mark Johnson (2006), Contextual dependencies in unsupervised word segmentation, in *In International Conference on Computational Linguistics and Association for Computational Linguistics (COLING/ACL*, (673–680).

Hafer, M. A. & S. F. Weiss (1974), Word segmentation by letter successor varieties, *Information Storage and Retrieval* 10:371–385.

Harman, Donna (1991), How effective is suffixing, *Journal of the American Society for Information Science* 42:7–15.

Harris, Zellig (1955), From phoneme to morpheme, *Language* 31:190–222.

Harris, Zellig (1967), Morpheme boundaries within words: Report on a computer test, *Transformations and Discourse Analysis Papers* 73.

Harris, Zellig S. (1991), *A theory of language and information: a mathematical approach*, Clarendon Press ; Oxford University Press, Oxford [England] New York, 90039472 Zellig Harris. Includes bibliographical references and index.

Hull, David A. (1996), Stemming algorithms: A case study for detailed evaluation, *Journal of the American Society for Information Science* 47:70–84.

Johnson, C. Douglas (1972), *Formal Aspects of Phonological Description*, Mouton, The Hague.

Johnson, Mark (2008), Using adaptor grammars to identify synergies in the unsupervised acquisition of linguistic structure, in *Proceedings of ACL-08: HLT*, Association for Computational Linguistics, Columbus, Ohio, (398–406).

Johnson, Mark, Thomas L. Griffiths, & Sharon Goldwater (2006), Adaptor grammars: A framework for specifying compositional nonparametric bayesian models, in Bernhard Schölkopf, John C. Platt, & Thomas Hoffman (eds.), *NIPS*, MIT Press, ISBN 0-262-19568-2, (641–648).

Kaplan, Ronald M. & Martin Kay (1981), Phonological rules and finite-state transducers, Linguistic Society of America, New York.

Karttunen, Lauri (1993), Finite-state constraints, in John Goldsmith (ed.), *The Last Phonological Rule: Reflections on Constraints and Derivations,*, University of Chicago Press, Chicago, (173–194).

Karttunen, Lauri & Kenneth R. Beesley (2005), Twenty-five years of finite-state morphology, in Antti Arppe, Lauri Carlson, Krister Lindn, Jussi Piitulainen, Mickael Suominen, Martti Vainio, Hanna Westerlund, & Anssi Yli-Jyr (eds.), *Inquiries into Words, Constraints and Contexts. Festschrift for Kimmo Koskenniemi on his 60th Birthday*, CSLI, (71–83).

Kazakov, Dimitar (2000), Achievements and prospects of learning word morphology with inductive logic programming, in *In Cussens and Dzeroski (Ed.), Learning Language in Logic*, (89–109).

Kazakov, Dimitar & Suresh Manandhar (1998), A hybrid approach to word segmentation, in *In Inductive Logic Programming: Proceedings of the 8th International Workshop (ILP-98)*, Springer, (125–134).

Kieffer, John C. & En hui Yang (2000), Grammar based codes: A new class of universal lossless source codes, *IEEE Transactions on Information Theory* 46:2000.

Kit, Chunyu (2000), *Unsupervised Lexical Learning as Inductive Inference*, Ph.D. thesis, University of Sheffield.

Kit, Chunyu & Yorick Wilks (1999), Unsupervised learing of word boundary with description length gain, in *Proceedings of the CoNLL99 ACL Workshop*, (pages).

Kleene, S. C. (1956), Representation of events in nerve nets and finite automata, in Claude Shannon & John McCarthy (eds.), *Automata Studies*, Princeton University Press, Princeton, NJ, (3–41).

Klenk, Ursula (1994), Automatische morphologische Analyse arabischer Wortformen, *Beihefte der Zeitschrift fr Dialektologie und Linguistik* 83:84–101.

Klenk, Ursula & Hagen Langer (1989), Morphological segmentation without a lexicon, *Literary and Linguistic Computing* 4(4):247–253.

Kock, J. de & W. Bossaert (1974), *Introducción a la lingüística automática en las lenguas románicas*, volume 202 of *Estudios y Ensayos*, Gredos, Madrid.

de Kock, Josse & Walter Bossaert (1969), Towards an automatic morphological segmentation, in *International Conference on Computationa Linguistics: COLING 1969,*.

Koskenniemi, Kimmo (1983), *Two-level morphology: a general computational model for word-form recognition and production*, Ph.D. thesis, University of Helsinki.

Kraaij, Wessel & Ren Ee Pohlmann (1996), Viewing stemming as recall enhancement, in *In Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM Press, (40–48).

Krovetz, Robert (2000), Viewing morphology as an inference process, *Artificial Intelligence* 118(1-2):277–294.

Larkey, Leah S. (2002), Improving stemming for arabic information retrieval: Light stemming and co-occurrence analysis, in *In SIGIR 2002*, ACM Press, (275–282).

Li, Ming & Paul Vitányi (1993), *An introduction to Kolmogorov complexity and its applications*, Springer-Verlag New York, Inc., New York, NY, USA, ISBN 0-387-94053-7.

Mackay, David J. C. (2002), *Information Theory, Inference & Learning Algorithms*, Cambridge University Press, Cambridge.

Manandhar, Suresh, Sašo Džeroski, & Tomaž Erjavec (1998), Learning multilingual morphology with CLOG, in David Page (ed.), *ILP*, Springer, volume 1446 of *Lecture Notes in Computer Science*, ISBN 3-540-64738-4, (135–144).

de Marcken, Carl (1996), *Unsupervised Language Acquisition*, Ph.D. thesis, MIT.

Matthews, P. H. (1966), A procedure for morphological encoding, *Mechanical translation and Computational Linguistics* 9(1):15–21.

McCulloch, Warren S. & Walter Pitts (1943), A logical calculus of ideas immanent in neural activity, *Bulletin of Mathematical Biophysics* 5:115–133.

Medina Urrea, A. (2000), Automatic Discovery of Affixes by Means of a Corpus: A Catalog of SpanishAffixes, *Journal of Quantitative Linguistics* 7(2):97–114.

Medina Urrea, A. & J. Hlaváčová (2005), Automatic Recognition of Czech Derivational Prefixes, in *Proceedings of CICLing 2005*, Springer, Berlin/Heidelberg/New York, volume 3406 of *Lecture Notes in Computer Science*, (189–197).

Medina-Urrea, Alfonso (2006), Affix discovery by means of corpora: Experiments for spanish, czech, ralámuli and chuj, in Alexander Mehler & Reinhard Köhler (eds.), *Aspects of Automatic Text Analysis. Festschrift in Honour of Burghard Rieger*, Springer, Berlin / Heidelberg, volume 209 of *Studies in Fuzziness and Soft Computing*, (277–299).

Mooney, Raymond J. & Mary Elaine Califf (1996), Learning the past tense of english verbs using inductive logic programming, in *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, Springer-Verlag, London, UK, ISBN 3-540-60925-3, (370–384).

Nevill-Manning, Craig G. (1996), *Inferring Sequential Structure*, Ph.D. thesis, University of Waikato.

Nevill-Manning, Craig G. & Ian H. Witten (1997), Identifying hierarchical structure in sequences: a linear-time algorihtm, *Journal of Artificial Intellgence Research* 7(1):67–82.

Olivier, D.C. (1968), *Stochastic grammars and language acquisition mechanisms*, Ph.D. thesis, Harvard University.

Radhakrishnan, T. (1978), Selection of prefix and postfix word fragments for data compression, *Information processing and management* 14(2):97–106.

Rissanen, Jorma (2007), *Information and Complexity in Statistical Modeling*, Springer Publishing Company, Incorporated, ISBN 0387366105, 9780387366104.

Rissanen, Jorma & Eric Sven Ristad (1994), Language acquisition in the mdl framework, in *Language Computations, American Mathematical Society*, DIMACS, (149–166).

Ritchie, G.D., G.J. Russell, A.W. Black, & S.G. Pulman (1992), *Computational Morphology*, MIT Press.

Roark, Brian & Richard Sproat (2006), *Computational approaches to morphology and syntax*, Oxford University Press, Oxford.

Savoy, Jacques (2006), Light stemming approaches for the french, portuguese, german and hungarian languages, in *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, ACM, New York, NY, USA, ISBN 1-59593-108-2, (1031–1035), doi:http://doi.acm.org/10.1145/1141277.1141523.

Schone, Patrick & Daniel Jurafsky (2001), Knowledge-free induction of inflectional morphologies, in *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-01*, (1–9).

Shalonova, Ksenia & Peter Flach (2007), Morphology learning using tree of aligned suffix rules, in *Proceedings of the ICML-2007 Workshop on Challenges and Applications of Grammar Induction*.

Shannon, C. E. & W. Weaver (1949), *The Mathematical Theory of Communication*, University of Illinois Press, Urbana.

Sproat, Richard (1992), *Morphology and Computation*, MIT Press, Cambridge.

Sproat, Richard, Chilin Shih, William Gale, & Nancy Chang (1996), A stochastic finite-state word-segmentation algorithm for chinese, *Computational Linguistics* 22:377–404.

Teahan, W. J., Rodger Mcnab T, Yingying Wen T, & Ian H. Witten (2000), A compression-based algorithm for chinese word segmentation, *Computational Linguistics* 26:375–393.

Teh, Y. W., M. I. Jordan, M. J. Beal, & D. M. Blei (2006), Hierarchical Dirichlet processes, *Journal of the American Statistical Association* 101(476):1566–1581.

van den Bosch, Antal & Walter Daelemans (1999), Memory-based morphological analysis, in *In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, (285–292).

van den Bosch, Antal, Walter Daelemans, & T. Weijters (1996), Morphological analysis as classification: an inductive learning approach, in *Proceedings of NEMLAP 1996*, Ankara, (59–72).

Van den Bosch, Antal, E. Marsi, & A. Soudi (2007), Memory-based morphological analysis and part-of-speech tagging of Arabic., in G. Neumann A. Soudi & Antal van den Bosch (eds.), *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, Springer, (203–219).

Wolff, J.G. (1975), An algorithm for the segmentation of an artificial language analogue, *British Journal of Pscyhology* 66(1):79–90.

Wolff, J.G. (1977), The discovery of segments in natural langauge, *British Journal of Psychology* 68:97–106.

Wothke, Klaus & Rudolf Schmidt (1992), A morphological segmentation procedure for german, *Sprache und Datenverarbeitung* 16(1):15–28.

Xu, Jinxi & W. Bruce Croft (1998), Corpus-based stemming using co-occurrence of word variants, *ACM Transactions on Information Systems* 16:61–81.

Yarowsky, D. & R. Wicentowski (2000), Minimally supervised morphological analysis by multimodal alignment, in K. Vijay-Shanker & Chang-Ning Huang (eds.), *Proceedings of the 38th Meeting of the Association for Computational Linguistics*, Hong Kong, (207–216).

Zhang, Byoung-Tak & Yung-Taek Kim (1990), Morphological analysis and synthesis by automated discovery and acquisition of linguistic rules, in *Proceedings of the 13th conference on Computational linguistics*, Association for Computational Linguistics, Morristown, NJ, USA, (431–436).