Discovering Morphemic Suffixes
A Case Study In MDL Induction

Michael R. Brent, Sreerama K. Murthy [*] Andrew Lundberg

Abstract

This paper reports experiments in the automatic discovery of linguistically significant regularities in text. The minimum description length principle is exploited to evaluate linguistic hypotheses with respect to a corpus and a theory of the types of regularities to be found in it. The domain of inquiry in this paper is the discovery of morphemic suffixes such as English *–ing* and *–ly*, but the technique is widely applicable to language learning problems.

1 Introduction

Many recent papers have reported work on the automatic discovery of linguistic regularities in text. Most of these exploit statistics based on information theory to measure how likely two linguistic entities are to co-occur. Co-occurrence statistics have been used to assess semantic similarity [15], PP attachment preference [16], linguistically significant collocations [23], syntactic categories [4], and syntactic rules [5]. The above work raises two interesting questions: how should statistical measurements be interpreted, and how should measurements of different kinds of linguistic patterns be combined? This paper argues that structure discovery procedures based on the minimum description length (MDL) principle have a clear semantics within which measurements of distinct linguistic structures can be "naturally" combined to answer a single question. An MDL procedure for discovering morphemic suffixes illustrates the point. When run on the 1000 most common words in a sample of the Wall Street Journal, this procedure outputs *age, al, ed, ing, ion, ity, ly, ment, nce,* and *s.*

The use of statistical measurements raises two questions.

*Brent is in the cognitive science department, Johns Hopkins University. Murthy and Lundberg are in the computer science department. Address correspondence to: Sreerama K. Murthy, Department of Computer Science, Johns Hopkins University, Baltimore MD, 21218. Email: murthy@cs.jhu.edu

1. *How should statistical measures be interpreted?*
   Church (1988) and Hindle (1993) use co-occurrence statistics for lexical and syntactic disambiguation, so their numbers are interpreted operationally by disambiguation systems. But other researchers have collected numbers without providing any objective interpretation. For example, [12] describes a hierarchical cluster analysis on word-adjacency statistics. The authors claim that some of the thousands of resulting classes correspond roughly to familiar syntactic or semantic classes. But this interpretation of classes lacks semantics, operational or otherwise.

2. *How can measures of different linguistic regularities be combined?*
   Two intuitions that linguists use in identifying morphemes, such as English *-ing*, are (1) morphemic suffixes and stems recombine with one another to form multiple words; (2) morphemic suffixes provide information about the syntactic categories of words formed with them. Either one of these intuitions could be assessed by measuring the mutual information between suffixes and stems, or between suffixes and syntactic categories. But it is nontrivial to combine these two measurements. Arbitrary combinations, such as addition or multiplication, can yield meaningless numbers.

The MDL Approach Minimum description length (MDL) induction procedures [22, 17] have a generative semantics within which disparate information sources can be "naturally" combined. MDL induction has been successfully applied to a large number of learning problems in the past. Examples include hand-printed character recognition [14], decision tree induction [20], molecular evolution [1, 18], analysing dynamic systems [8], learning engineering models [21], clustering [6], computer vision [13] and constructive induction [19]. In the context of automated concept acquisition from linguistic corpora, however, applications of the MDL principle have been relatively few. MDL principle has been explored for lexical knowledge acquisition [2], speech segmentation [3, 7], and to phonology [10, 11].

Conceptually, induction can be viewed in terms of a module that enumerates a set of hypotheses, and another independent module that determines the most plausible hypothesis. MDL is a criterion for evaluating hypotheses in terms of how well they explain the regularities in the input. A hypothesis, or an *accounting* comprises of two components, namely, the *theory* and the *unexplained residual*. Among all accountings, the MDL principle prefers the one that is least stipulative, where stipulativeness is equated with information content. Typically, the problem of devising an encoding (for computing the information content) for an induction problem has two parts: (1) Devising a high level model of the process that generated the input data, and (2) Converting the model into a compact encoding.

The rest of this paper is organized as follows. The next section provides two models for the morpheme-discovery problem. The first makes use of the intuition that morphemes recombine to form multiple words. This model illustrates the intuitive appeal of MDL induction. The second model augments recombination with the intuition that words formed with a particular suffix tend to belong to particular syntactic classes. This model illustrates the way in which disparate information sources can be "naturally" combined within the MDL framework. Section 3 describes the specific encodings we use. Section 4 outlines the heuristics we used for searching the space of possible hypotheses. Section 5 describes some experiments on finding morphemic suffixes in word lists extracted from the Wall Street Journal. Section 6 concludes the paper.

2 Two Models

This section presents two models of the process responsible for generating the words in a corpus, along with the corresponding high-level representations of hypotheses.

2.1 Simple Recombination

Consider the problem of identifying morphemic stems and suffixes. Stems and suffixes can help account for the regularities in a corpus in terms of the following, admittedly simplistic, model of how the corpus was generated. The black box generating the words in the corpus has a finite list of stems and a finite list of suffixes, and it generates words by concatenating stems and suffixes freely. Given this free-concatenation model, the induction problem is to determine the stem list and the suffix list used in generating the words of a particular corpus.

For the sake of illustration, imagine that some corpus of English contained all and only the words listed under "Input Words" in Figure 1. Suppose that those words were generated by concatenating a stem from a stem list and a suffix from a suffix list. Then the two lists shown at right in Figure 1 constitute a plausible hypothesis about the lists used to generate the corpus. The epsilon on the suffix table stands for the empty string, which allows the strings on the stem table to occur without any suffix. The two lists constitute the theory portion of an accounting. This theory predicts that any word formed by concatenating one element from each list is a legitimate word of English. However, the theory fails to predict that, among all the words that can be generated from these stems and suffixes, a particular subset does not occur, including *dumped, dumping, preferentis*, and *preferentied.* Such incorrect predictions are inevitable here, since the free–concatenation model ignores a many morphological facts. Under the theory, the failure of these predicted words to occur in the input sample is a matter of unexplained residual.

The stem and suffix lists, written in alphabetic characters, constitute high–level encodings of the theory. A high-level encoding of the unexplained residual would specify which words, among those that the theory predicts, are attested

Input Words

walk	referral
walks	refer
walked	refers
walking	dump
referred	dumps
referring	preferential

Stem Table

stem	code
walk	1
referr	2
refer	3
dump	4
preferenti	5

Suffix Table

suf.	code
ϵ	1
s	2
ed	3
ing	4
al	5

Encoded Words

stem	suf.	stem	suf.
1	1	2	5
1	2	3	1
1	3	3	2
1	4	4	1
2	3	4	2
2	4	5	5

Figure 1: An input lexicon (left) and a generative explanation for it (right).

in the corpus, and, by process of elimination, which are not. The table titled "Encoded Words" in Figure 1 is one possible representation of the residual. The stem table specifies the correspondence between stems and stem code words, and the suffix table does the same for suffixes. In this high–level code, the codewords are simply the ordinal indices of the stem and suffix on their respective tables.

The representation shown in figure 1 illustrates the intuitive appeal of the MDL principle. Although the binary encoding shown in the next section constitutes a more accurate evaluation, the relative information content of different accountings can be approximated by counting the number of characters in their high-level representations. Now, consider an alternative theory within the free-concatenation model. In the stem and suffix tables shown at left in Figure 1, consider dividing the words in the input sample in such a way that each hypothesized suffix is *extended* by one character, while each hypothesized stem is *reduced* by one character. In this case, the total number of letters in the stem and suffix tables clearly exceeds the total of the theory in Figure 1. This accounting stipulates more information than the original instance and also makes less accurate predictions about English words.

2.2 Combining Multiple Measures

Consider an approach to discovering morphemic suffixes that combines information from two sources. The first source is just as before — stems and suffixes

A. Input Lexicon

walk	V	referring	V
walk	N	referring	A
walks	V	referral	N
walks	N	refer	V
walked	V	refers	V
walking	V	dog	N
walking	N	dogs	N
referred	V	preferential	A

B. Category Table

suf.	cat.	code
ϵ	V	0
	N	1
s	V	0
	N	1
ed	V	0
ing	V	0
	N	1
	A	2
al	A	0
	N	1

C. Encoded Lexicon

1	1	0	2	4	0
1	1	1	2	4	2
1	2	0	2	5	1
1	2	1	3	1	0
1	3	0	3	2	0
1	4	0	4	1	1
1	4	1	4	2	1
2	3	0	5	5	0

Figure 2: Input lexicon, category table, and encoded lexicon for the class of induction of generators that label words with syntactic categories.

recombine into multiple words. The second source is the tendency of morphemic suffixes in English to determine the syntactic category of words formed with them. For example, words bearing the morphemic suffix *-s* are likely to be verbs or nouns, but unlikely to be adjectives or adverbs.

Suppose that the input to the suffix identification procedure consists of words along with their major syntactic categories, as shown under "Input Lexicon" in Figure 2. Ambiguous words occur in the input once for each of their possible categories. The following model exploits both sources of information: words are generated by selecting a stem and a suffix from their respective lists, and then selecting a syntactic category from a list of syntactic categories *available to words with the selected suffix.* A particular theory in this framework consists of a stem list, a suffix list, and a list of syntactic categories available for words with each suffix. For example, given the input shown in Figure 2, a plausible theory includes the stem and suffix lists from Figure 1, along with the category table in Figure 2. The fewer categories each suffix can take, the smaller the category table, and hence the more highly rated the accounting, all other things

stem	suf.	stem	suf.
00	00	01	110
00	01	100	00
00	100	100	01
00	101	101	00
01	100	101	01
01	101	1100	110

Figure 3: Encoded lexicon for the hypothesis in Figure 1

being equal.

In the combined model, the unexplained residual is the particular set of stem-suffix-category triplets that are observed in the sample, among all those triplets that the hypothesis permits. The encoded lexicon table in Figure 2 is one possible representation of this residual. (Note that the category indices in the table are relative to the suffix.)

It should be noted how easily information from such disparate sources as orthography and lexical syntax can be combined in MDL induction.

3 Encoding

The representation of tables shown so far, which includes alphabetic characters, numerals, and alignment into rows and column, is less than ideal for measuring information. One reason is that the information communicated by alignment in these tables is not included in the character-counting measure. More important, the tables contain unnecessary characters, and therefore they fail to show each hypothesis in the best light. We need to render (encode) the tables shown above as a single sequence of zeros and ones that contain much less redundancy. The encoding must be a one-to-one transformation.

The encodings in this paper use binary sequences, called code words, for stem codes, suffix codes, category codes, and for individual letters. The character counting measure of information is replaced by a continuous approximation to bit count. The items that occur most frequently get the shortest bit encodings. An example is shown in Figure 3, where the sequential decimal code words of Figure 1 have been replaced by binary code words assigned on the basis of frequency.

The Shannon-Fano (SF) code is a near-optimal method of determining the length of each code word in terms of its frequency. The length of the SF code words is approximately the binary logarithm of the inverse of their relative frequency. Each stem and suffix in Figure 3 is an SF code.

111. . .10	l([walk]) l([referr]) . . .	[walk] [referr] . . .
$1 + \max_{S \in \text{stems}} l([S])$	$\lvert\text{stems}\rvert \times \max_{S \in \text{stems}} l([S])$	$\sum_{S \in \text{stems}} l([S])$

111. . .10	l(walk) l(referr) . . .	[w][a][l][k][r][e][f][e][r][r] . . .
$1 + \max_{S \in \text{stems}} l(S)$	$\lvert\text{stems}\rvert \times \max_{S \in \text{stems}} l([S])$	$\sum_{C \in \text{chars}} f_{\text{stem}}(C) l([C])$

Figure 4: Binary encoding of a stem table

Let the notation $[S]$ refer to the code word for S, and $l([S])$ to the length of that code word in bits. Let $f(S)$ be the frequency with which the code word for stem S occurs in the Encoded Lexicon table. Then its total frequency, counting the occurrence in the stem table, is $f(S)+1$, and its relative frequency is $\frac{f(S)+1}{\sum_{S' \in \text{stems}} f(S')+1}$, so the length of its SF code word is

$$l([S]) = \log_2 \left(\frac{\sum_{S' \in \text{stems}} (f(S')+1)}{f(S)+1} \right) \tag{1}$$

The code word for stem S occurs $f(S)+1$ times in the representation, so the total number of bits used to encode those occurrences is: $(f(S)+1)l([S])$ Summing over all stems, the total bits used for all stem code words is: $\sum_{S \in \text{stems}} (f(S)+1)l([S])$

Since the tables are represented as an unbroken string of zeros and ones, some mechanism must be provided for determining where one code word ends and the next one begins. The Shannon-Fano lengths make it possible to select a set of code words such that no code word is a prefix of another code word. Given such a set of code words, it is possible to tell where one ends and the next one begins in any sequence, if and only if the set of code words is known to the decoder. While the stem and suffix tables are being decoded, however, the code words are still unknown. As a result, some additional mechanism is needed to indicate their boundaries. One such mechanisms is an array specifying the number of bits in each code word. This is shown inside the rectangle in the top half of Figure 4.

[walk] refers to the code word for *walk*, and l([walk]) refers to its length in bits. The representation actually consists of three segments. The rightmost segment contains the code words for each of the stems, in sequence, with no delimiters. The middle segment consists of a sequence of binary integers, each of which gives the length of the corresponding code word—the first integer gives length of the first code word, etc. The lengths of the code words for stems are all equal to $\max_{S \in \text{stems}} l([S])$. This length is specified as a unary integer in the first segment. Together, these three segments constitute a complete encoding of

the right column of the stem table.

The Shannon-Fano code can also be used to assign frequency-based code words to individual letters. This requires including another table that gives the keys to the letter code. Assuming that a letter code table is included, the left column of the stem table can be encoded as shown in the bottom half of Figure 4.

This encoding is divided into three segments much like those of Figure 4. The length of each segment is shown below it, where $f_{stem}(C)$ is the frequency of character C in the stem table, l(S) is the length of stem S in letters, and |stems| is the number of stems in the stem table.

The same principles are used to construct encodings for the suffix table and the letter code table. Putting together the lengths of all these encodings, we get a formula for the continuous approximation to the number of bits needed to represent a hypothesis. Among all the accountings in the simple-concatenation model that our system explores, it selects the one that minimizes the value of this representation–length. For the model using syntactic categories, several more terms of the same sort are added to account for the information content of the category table.

4 Search Heuristics

The encoded words table in the representations described above specifies a splitting of each word in the input into two parts, one on the stem table and one on the suffix table. The number of possible splits of the input is equal to the product of the lengths of all the words in it. Clearly, it is not possible to evaluate every single accounting.

The first heuristic we use to reduce the number of accountings is a search strategy that first determines the suffix table, and *then* chooses a stem table and an encoded lexicon. This strategy evaluates one hypothesis for each different stem table. This strategy is implemented by ensuring that, given a suffix table T, every word that ends in a string $S \in T$ is split before S. To ensure determinism, we considered only those suffix sets in which no (non-empty) suffix was a suffix of another.

Evaluating only one hypothesis for each suffix set reduces the search space substantially, but the number of possible suffix sets still prohibits exhaustive search. To make search tractable, all the word-final letter sequences in the lexicon are ranked according to the ratio of the relative frequency of the sequence divided by the relative frequencies of its component letters. For example, if c is the total number of characters in the input, the rank of *-ing* would be $\frac{f(ing)/(c-2)}{f(i)f(n)f(g)/c^3}$. We use greedy search, that tries adding candidate suffixes, one at a time, from highest rank to lowest. (Initially the suffix set is empty.) In some cases, adding a suffix entails discarding one or more other suffixes. When the description length can no longer be reduced by adding a suffix, the system tries removing one suffix at time to see if that helps. This addition-removal process is repeated till there is no improvement in description length.

5 EXPERIMENT

The evaluation function and search techniques described above were tested on input lexicons of various sizes, prepared from a sample of the Wall Street Journal tagged for part-of-speech by the Penn Treebank project. All words except those containing capital letters or non-alphabetic symbols were sorted by frequency, and input lexicons of different sizes were prepared by taking the most frequent words from the top of the sorted list. Experiments were done using both the simple concatenation model and the combined model. For the combined model, the Penn categories were mapped down to a set of five categories representing common nouns, verbs, adjectives, adverbs, and all other words.

The best results were obtained with lists of the 1,000 and 2,000 most frequent words. First consider the combined model. On the 1,000 word input the hypothesis with the best evaluation had the following suffixes on its suffix table: *age al ed ing ion ity ly ment nce* and *s*. *nce* is reasonable, since it is the orthographic sequence common to a morphological process that yields either *ance*, as in *guidance*, or *ence*, as in *preference*. For the 2,000 word input the best hypothesis used all the suffixes from the 1,000 word lexicon, plus the following: *able ary ful ive ld ncy one out ship* and *sure*. Of these, only *ld* is a meaningless final string. When the simple concatenation model was used, the 1,000 word input yielded three incorrect suffixes in addition to the ten correct ones found by the other method: *me, st,* and *ve*. This is in accord with the expectation that using more evidence should yield better results. On the 2,000 word input, however, the difference between the two evaluations was only a single error, *wn*.

The general trend was that increasing the number of input words (and hence decreasing the average frequency) led to finding more correct free morphemes, like *ball*, and also more incorrect final strings. Many of the incorrect strings were extensions of correct morphemes, such as *ional* and *gical*. Figure 5 summarizes the results as a function of the input-size/frequency variable.

6 Discussion and Conclusions

Our algorithm appears to have successfully exploited the two linguistic intuitions that morphemes recombine and that suffixes predict the syntactic categories of words formed with them. Of the two intuitions, recombination was clearly more significant than category prediction in these experiments. Category prediction improved accuracy significantly on small inputs, but its positive effect eroded steadily with increasing input size. At 4,000 and 8,000 words it showed a slight trend toward reducing accuracy.

Words	B	EB	F	EF	E	Tot	%M	%P
500	6/7				10/3	16/10	38/70	38/70
1000	10/10				3/0	13/10	78/100	78/100
2000	15/16	2/1	2/2		1/1	20/20	95/95	85/90
4000	19/18	7/7	8/8	1/1	4/5	39/39	90/87	70/67
8000	27/28	9/9	19/18		3/6	58/61	95/90	79/75

Figure 5: Categorization of suffixes output as a function of the number of words in the input lexicon. Syntactic category method / plain recombination method. B = Bound morpheme; F = Free morpheme, E = Error. L designates extensions. Tot = total number of suffixes hypothesized. %M = (Tot - E) as a percentage of Tot. %P = (B + F) as a percentage of Tot.

We also observed that the rate of increase of the number of morphemes was much less than that of the lexicon size. When the input increased from 500 to 8000 words, a factor of 16, the number of morphemes hypothesized only increased by a factor less than six. In addition, the increase in number of morphemes when the lexicon size increased from 4000 to 8000 was much less than that from 1000 to 2000 or 2000 to 4000, suggesting a trend toward convergence. The number of correct bound morphemes increased even more slowly — approximately as the logarithm of the input size. The growth in the number of extensions of bound morphemes and in free morphemes was more pronounced than that in the number of plain errors.

In part, these results reflect the fact that the intuitions encoded here do not provide a complete account of what is meant by the notion morpheme. Other factors include: the syntactic category of the stem predicts which morphemes will attach to it; stems appear as independent surface forms; suffixes can cause regular changes in the orthography of the composite word; morphemes generally show some sign of productive use; and both stem and morphemic suffix have meaning. Given that the algorithm presented here ignores all those factors, it does remarkably well at discovering morphemic suffixes. More importantly, the algorithmic induction approach or the generative approach provides a framework within which most of these intuitions can be encoded and tested. In addition, algorithmic induction promises to be a useful tool for refining and making explicit the linguistic intuitions themselves.

One key issue for the success of any MDL-based induction is search. Nontrivial linguistic theories have so many degrees of freedom that exhaustive search is impossible. Discovering powerful domain dependent heuristics is where activity should be concentrated.

References

[1] Lloyd Allison. Minimum message length encoding, evolutionary trees and multiple alignment. Technical report, Department of Computer Science, Monash University, Clayton, Victoria, Australia, 1991.

[2] Michael R. Brent. From grammar to lexicon: Unsupervised learning of lexical syntax. *Computational Linguistics*, 19:243–262, 1993.

[3] M.R. Brent, A. Gafos, and T.A. Cartwright. Phonotactics and the lexicon: Beyond bootstrapping. In E. Clark, editor, *Proceedings of the 1994 Stanford Child Language Research Forum*, Cambridge, UK, 1994. Cambridge University Press.

[4] Eric Brill, David Magerman, Mitchell Marcus, and Beatrice Santorini. Deducing linguistic structure from the statistics of large corpora. In *Proceedings of the DARPA Speech and Natural Language Workshop, June 1990*, pages 275–282, Harriman, NY, 1990. DARPA.

[5] Eric Brill and Mitchell Marcus. Automatically acquiring phrase structure using distributional analysis. In *Darpa Workshop on Speech and Natural Language*, Harriman, NY, 1992. DARPA.

[6] P. Bryant. On detecting clusters using the MDL principle. In *Proceedings of the 4th Conference of the International Federation of Classification Societies*, pages 154–155, 1993.

[7] T.A. Cartwright and M.R. Brent. 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*, pages 83–91. Association for Computational Linguistics, 1994.

[8] R. M. Chen. State and parameter estimation for dynamic systems with colored noise using MDL. *IEEE Trans on Automatic Control*, 36:813–823, 1991.

[9] K. Church. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Proceedings of the 2nd ACL Conference on Applied NLP*. ACL, 1988.

[10] T. Mark Ellison. Discovering planar segregations. In *AAAI Spring Symposium on Machine Learning of Natural Language and Ontology*. AAAI, 1991.

[11] T. Mark Ellison. The iterative learning of phonological rules. *Computational Linguistics*, 1994. Forthcoming.

[12] Steven Finch and Nick Chater. Bootstrapping syntactic categories using statistical methods. In Walter Daelmans and David Powers, editors, *Background and Experiments in Machine Learning of Natural Language: Proceedings of the 1st* SHOE *Workshop*, 1992.

[13] Noah S. Friedland. *Utilizing Energy Function and Description Length Minimization for Integrated Delineation, Representation and Classification of Objects.* PhD thesis, Department of Computer Science, University of Maryland, College Park, 1993.

[14] Q. Gao and M. Li. The minimum description length principle and its application to online learning of handprinted characters. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 843–848, Los Altos, CA, 1989. Morgan Kaufmann.

[15] D. Hindle. Noun classification from predicate argument structures. In *Proceedings of the 28th Annual Meeting of the ACL*, pages 268–275. ACL, 1990.

[16] Don Hindle and Mats Rooth. Structural ambiguity and lexical relations. *Computational Linguistics*, pages 103–120, 1993.

[17] Ming Li and Paul M. B. Vitányi. Inductive reasoning and kolmogorov complexity. *Journal of Computer and System Sciences*, 44:342–384, 1992.

[18] A. Milosavljevic and J. Jurka. Discovery by minimal length encoding: A case study in molecular evolution. *Machine Learning*, 12:69–89, 1993.

[19] B. Pfahringer. Controlling constructive induction in ciPF: An MDL approach. In F. Bergadano and L. de Raedt, editors, *Proceedings of the European Conference on Machine Learning*, pages 242–256, Berlin, 1994. Springer.

[20] J. Ross Quinlan and Ronald L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248, 1989.

[21] R. B. Rao and S.C. Lu. Learning engineering models with the minimum description length principle. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 717–722, Menlo Park, CA, 1992. AAAI Press/MIT Press.

[22] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[23] Frank Smadja. Retrieving lexical collocations from text: Xtract. *Computational Linguistics*, pages 143–177, 1993.