Morphological Induction Through Linguistic Productivity

Sarah A. Goodman
University of Maryland-College Park
sagoodm@umd.edu

Abstract

The induction program we have crafted relies primarily on the linguistic notion of `productivity` to find affixes in unmarked text and without the aid of prior grammatical knowledge. In doing so, the algorithm unfolds in two stages. It first finds seed affixes, to include infixes and circumfixes, by assaying the character of all possible internal partitions of all words in a small corpus no larger than 3,000 tokens. It then selects a small subset of these seed affixes by examining the distribution patterns of roots they fashion to, as demonstrated in a possibly larger second training file. Specifically, it hypothesizes that valid roots take a partially overlapping affix-set, and develops this conjecture into agendas for both feature-set generation and binary clustering. It collects feature sets for each candidate by what we term affix-chaining, delineating (and storing) a path of affixes joined, with thresholding caveats, via the roots they share. After clustering these resultant sets, the program yields two affix groups, an ostensibly valid collection and a putatively spurious one. It refines the membership of the former by again examining the quality of shared root distributions across affixes. This second half of the program, furthermore, is iterative. This fact is again based in productivity, as we ration that, should a root take one affix, it most likely takes more. The code therefore seeds a subsequent iteration of training with affixes that associate with roots learned during the current pass. If, for example, it recognizes *view* on the first pass, and *viewership* occurs in the second training file, the program will evaluate `-ership`, along with its mate `-er`, via clustering and root connectivity on the second pass. The results of this method are thus far mixed according to training file size. Time constraints imposed by shortcomings in the algorithm's code have thus far prevented us from fully training on a large file. For Morpho Challenge 2008, not only did we only train on just 1-30% of the offered text, thereby saddling the stemmer with a number of Out Of Voculary items, but, we also divided that text into smaller parts, thereby, as the results show, omitting valuable information about the true range of affix distributions.

Keywords

productivity, feature sets, clustering, infixation

1 Statement of Problem

Here we offer an algorithm that can detect not only standard prefixation and suffixation, but also multiple infixes per word, covering templatic morphologies; reduplication, to include a combination of reduplication and infixation; and point-of-affix gemination. Furthermore, the program can elicit these linguistic features from a relatively small file. As it largely eschews calculations based on root or affix frequency in a training file, it can therefore avoid sparse data pitfalls. It may therefore be a particular boon to processing of Less Commonly Taught Languages, given that text-based resources are often hard to find in certain regional languages.

To shepherd this algorithm to such success, we will gradually and successively winnow affix candidates through two general principles, one founded in a breed of common sense, the other in descriptive linguistic doctrines. First, we repeatedly employ the use of what we term `co-occurrence`, a form of boot-strapping. In this, we assume that a less reliable unit, be it a character or an affix, in the context of a more trustworthy counterpart should be retained in the candidate pool. Here, this context can refer to a single word or to a root taken in common by multiple affixes. Second, we leverage and deploy co-occurrence by placing great faith in the linguistic notion of `productivity`. The term ultimately refers to the prevalence of a linguistic unit, generally a morpheme, in a language. In so denoting, it specifically invokes the degree to which a unit may combine or directly co-occur with a separate linguistic unit. We interpret the prospect of modified linguistic ubiquity to increase recall, especially among derivational affixes, by specifically investigating root connectivity among affix candidates.

2 Previous Work

Dasgupta and Ng (2007) develop a knowledge-free analyzer to parse Bengali by relying on perceived general principles of morphemic concatenation and root distribution, and they assess them by means of heuristics that concern frequency but mostly stop short of being probabilities.

Càvar *et. al* (2004) attempt an induction schema based on a combination of Dasgupta and Ng's seeding techniques to derive a weighted implementation of information theory metrics. The authors generate a seed list of morphemes substring matching and refine the resultant candidate list by incremental construction of a grammatical framework meant to dictate, or at least clarify, emergent patterns in the input.

Brent *et. al* (1995) and Goldsmith (2001) both practice Minimal Description Length and take the condensation of space quite seriously. Their programs find affixes based on how well they can shorten the training corpus by counting repeated endings and beginnings of words just one time. To ensure that each doesn't over-estimate the presence of affixes by this process, the method imposes a weight or penalty for each affix guessed, partially based upon its length.

Monson, et. al (2007) structure the backbone of their induction program to be what they term *schemes*, which are similar, if not identical to, Goldsmith's signatures. These consist of classes of affixes united by a set of roots that exhaustively take them. Each such scheme they arrange into a lattice network based upon parent-child relations. The authors then traverse this constructed network and target possible spurious classes based largely upon their lack of a parent-child stem ratio above a certain, variable threshold. They then engage in group agglomerative hierarchical clustering to further target valid members of the remaining schemes.

Bernhard (2005) and Dejean (1998) develop a means of segmentation based on word-internal character transition probabilities. Their concept is, in fact, the basis of Harris (1955), and both modern instantiations of the idea, perhaps as expected, vary slightly from each other. Dejean, for his part, uses a bootstrapping measure to boost the discovery or morphemes based upon patterns already heavily favored as correct. In Bernhard's conceptualization of the solution, transition probabilities are employed based upon substring incidence ratios.

Shortcomings of the above approaches include a requirement for large amounts of training data (Dasgupta and Ng; Ćavar *et. al*), increased memory accommodations (Monson *et al*), low recall (MDL approaches; Dasgupta and Ng) and general over-reliance on mathematical technique at the expense of linguistic primacy (Brent and Goldsmith).

3 Generating Seed Affixes

The algorithm's first section is guided by two goals: first, we wish to produce infixes among prefixes and suffixes, and, second, we wish to yield multi- morphemic combinations of concatenative languages amid the sparse data given by smaller corpora. Here, the code generates what we'll call `seed affixes`. These are character sequences assigned a position relative to either the beginning or ending of a word and are selected based upon their frequency of occurrence in a relatively small file of approximately 3000 tokens.

The first step in this process is to take all possible internal partitions of every word in the seed file, which themselves sum to $2^{word.length-1}$. We call each resultant partition analysis a *partition schema*. Some of these schemas we then toss from consideration, both to manage memory allocation and, primarily, to downplay the significance and effect of such unigrams. As a rule, we omit any partition schema, such as any of the 16 listed in table 1, if the number of internal candidate morpheme breaks for a given word is greater than half the letters composing that word.

Potentially, each break in each schema represents a morpheme. We reduce this multitude in two stages. The first method examines the frequency of each positioned segment across partition schemas and yields two sets of 100 candidates, one set for potential prefixes and infixes and a second for suffixes and additional infixes. This task modifies the assessed prevalence of each segment according to a weight ratio it applies to each instance, dividing the longest word length in the seed corpus by the length of the word containing the segment being evaluated. Such a

t|iger	t|i|ger	t|ig|er	ti|ger
t|i|g|er	t|ige|r	tig|er	tige|r
tiger	tig|e|r	ti|ge|r	ti|g|er
ti|g|e|r	t|ig|e|r	t|i|ge|r	t|i|g|e|r

Table 1: All Possible Internal Partitions of *tiger*

tiger	tige|r	tig|e|r	tig|er
ti|ger	ti|ge|r	ti|g|er	t|iger
t|ige|r	t|ig|er	t|i|ger	

Table 2: All Retained Internal Partitions of *tiger*

measure avoids disproportionately overstating the effect of longer words in the seed file, which, by their nature, propagate a greater number of partition schemas. The second stage uses each list of 100 positioned segments to deliver all possible segmentations of the seed corpus. It then examines each (possibly competing) lexical analysis to derive a smaller set of candidates: those that that are the 15 most frequent segments and the less frequent segments that occur along side them in anyone analysis. This generally yields between 40-50 seed affix candidates.

The final step in seed affix generation involves taking all possible sequential or quasi-sequential combinations of the the top 15 affix candidates and their bootstrapped associates. This process covers, or, at least supplements, affix generation more languages that one might think, as the seed segments themselves are usually no longer than two characters. We can therefore derive the English affix *ous* from seed affixes *s|0* and *ou|1*. This combinatory process is even more important for polysynthetic languages. In these cases, individual morphemes can occur in multiple contexts of others, and, theoretically, the previous work done on partition schemas should discover these individual units separately. Taking their combinations, then, allows us to generate the range of concatenative manifestations that the particular language shows, even if only a fraction of them were present in the seed file.

4 Using a Trie

In order to assess the validity of the seed affix candidates and their possible combinations, we store contents of a second, usually larger, file into both a forwards and backwards trie. We chose a trie to winnow good from bad sequences for simple reasons of efficiency. Such a storage mechanism reduces memory load because it conflates stored values for words composed of identical sub-sequences. A trie also allows for relatively quick root recovery without reliance on regular expressions.

In addition to simply weeding out invalid or unattested candidate affixes, through simple book-keeping of sorts, the trie can also be used to learn affix candidates based on, or grown from, currently attested segments. This process is similar to the deduction structures of Harris (1955), which itself inspired Dejean (1998). Their system, like ours, functions on character succession options following a given segment that partially enumerates a word or words in a language. If the range of valid alphabetic options extending the sequence is above a certain threshold, it may signal a morpheme boundary. In our version, we admit extensions based both upon whether a ratio-metric is passed and upon whether the sequence has initiated a new branch in the trie. The latter method, in particular, allows supersets of unsuccessful subsequences to be retained. For example, the sequence *me* grown from the base affix *nt* may fail to meet the ratio specifications, but its follow-on sequence *eme* might not.

Rank	Segment	Score	Rank	Segment	Score	Rank	Segment	Score
0.	e|0	0.35	1.	s|0	0.31	2.	e|1	0.23
3.	d|0	0.22	4.	i|1	0.20	5.	a|1	0.20
6.	n|1	0.20	7.	y|0	0.20	8.	t|0	0.19
9.	n|0	0.19	10.	g|0	0.18	0.	ed|0	1
1.	ng|0	0.96	2.	ing|0	0.85	3.	er|0	.52
4.	es|0	0.50	5.	on|0	0.49	6.	in|1	0.49
7.	ion|0	0.36	8.	nt|0	0.36	9.	al|0	0.35
10.	ti|1	0.26	11.	re|0	0.25	12.	re|2	0.25
13.	re|1	0.25	14.	le|0	0.24	15.	ry|0	0.24
16.	en|0	0.24	17.	ly|0	0.23	18.	ce|0	0.23
19.	te|0	0.23	20.	ent|0	0.23	21.	er|1	0.22
22.	an|1	0.21	23.	ar|1	0.20	24.	te|1	0.19
25.	ca|1	0.19	26.	ll|0	0.19	27.	en|1	0.19
28.	ca|2	0.19	29.	on|1	0.19	30.	ve|0	0.18
31.	ty|0	0.18	32.	th|1	0.16	33.	ti|2	0.16
34.	it|1	0.16	35.	ts|0	0.16	36.	st|0	0.15
37.	st|2	0.15	38.	co|2	0.15	39.	se|1	0.14
40.	li|1	0.14	41.	ou|1	0.13	42.	he|1	0.13
43.	in|2	0.13	44.	al|1	0.13			

Table 3: Final List of English Positioned Suffix Segments and Their Relative Scores

5 Using Root and Affix Distributions

The algorithm now separates good from bad affix candidates by utilizing two parallel vistas: the perspective of individual roots and the affix sets each takes, and the converse examination of individual affixes and the root sets each assumes. In this, it determines patterns of diverse reliability though co-occurrence. Specifically, the program assumes that valid roots take a partially overlapping set of affixes in common, and it creates two sets based upon this choice. It then utilizes set membership in conjunction with thresholding caveats to further extend validity assessments, such that, if there's evidence that affix α is a true affix in a language, and affix β occurs on a subset of roots that α does, then it takes steps to consider β a true affix as well.

5.1 Clean-Up Procedures

Before examining distribution patterns across the present set of roots and affixes, we potentially manipulate each slightly, either expanding them or over-writing them. First, many seed affix candidates resisted combination with other early induced units and are therefore somewhat short. While many of these may be correct, the pool is still largely incomplete. We expand the affix collection, then, to increase the overall membership and enhance the prospect of greater overlap in clustering, by taking a Maximum Likelihood Estimate (MLE) of root termination (for suffixes) or initiation (for prefixes) patterns for each root. Here, we consider up to the first or last three character and retain those extensions scoring above a certain threshold.

Secondly, our code looks for sequences of contiguous or non-contiguous reduplication between root and affix. This type of linguistic phenomenon must be addressed in our case as it evokes the same trepidation involving the hypothetical training pair *stars* and *started*. In this case, the training pair may be *shops* and *shopping*, where the root taking a both a valid and an invalid affix supports both guesses. In these cases, the algorithm changes the affix's repeated sequences to the cover symbol `RR`, for reduplication, or `R`, point of change gemination, if it can find other instances to apply the same revisions. If both *planning* and *occurring* exist in the training file, for instance, then the program creates an affix class of *Ring*. For the Tagalog pairs *binabasa* and *sinasabi*, the

program creates the affix class *RinR* and assigns it the roots *basa* and *sabi*.

5.2 Feature Set Generation in Preparation for Binary Clusterization

We now develop feature sets based upon patterns of root co-occurrence. The backbone of this process is therefore root productivity, and it was selected for simple predictability. Valid roots in a language take a wider set of affixes in common than invalid roots, and, because of this, the expected outcome, largely borne out in test files, is that feature sets of valid affixes will show greater membership and a higher degree of content overlap. To produce feature sets that mimic this theory, we co-opt the notion that roots can take multiple affixes and enact that fact into a chain-based agenda. Specifically, for each base affix, we examine affixes with which it shares a common root and use these to potentially populate its feature set. We call these *degree-1 affixes*. Not content to leave a feature set at that, seeking instead to broaden internal similarity across sets, we further examine the root distribution patterns of the degree-1 affix. We now potentially add to the same set any affix that shares a root in common with a degree-1 candidate. Call these affixes *degree-2 affixes*.

For this desired result to be reliably wrought, thresholds must be in place to block incidental feature chaining. For instance, if *started* and *stars* both exist in the training file, then *ted|0* is a potential degree-1 affix for base affix *s|0*, despite the later being valid and the former being less so. We therefore resort to measuring the incidence strength between a base and a degree-1 affix by tallying their root connectivity. We call this a primary threshold. In measuring the relatability between a base affix and a degree-2 affix, on the other hand, we examine the cardinality of the degree-1 population that lead us to the association. We must see a certain number of *affixes* that link *s|0* and *ting|0* beyond simply *ted|0*. We call this threshold a secondary threshold.

The exact values of both the primary and secondary thresholds depend on both the number of types in the test file and which application of feature set generation we find ourselves on. The feature chaining content is in fact run three times with three separate combinations of primary and secondary thresholds, with goal of each iteration to increase the size of a given feature set. The first run includes primary and secondary thresholds that are based on type count; if there are less than 1000 types in the corpus, both thresholds are 2, otherwise, they're .0015 times the number of types. The second application includes the same primary threshold, but a no secondary threshold. The second condition was voided under the assumption that primary thresholds serve as a gateway, and, if they're good enough, they should allow enough trusted content through to negate the use of further safeguards. Unfortunately, for many affixes, both valid and spurious, the primary threshold may be too high to allow most content to pass and many feature sets are left with only the base affix itself and potentially one other member. The third iteration targets only those affixes and instead dictates a primary threshold reduced by half, to be a minimum of 2, and no secondary threshold.

5.3 Binary Clusterization

We next engage in binary clusterization, using affix-populated feature-sets to define a group of valid affixes against a group of, potentially, spurious choices. Grouping as such punctuates each of the three applications of feature set generation. We define therefore sets, then seek to define groups based upon those sets. Furthermore, we carry over results from previous groupings to subsequent applications; the first clustering iteration yields two clusters, while the second two rounds, making use of increasingly expanded feature sets, grows the content of each. During this process, we also avoid repetition of competent categorization. Affixes admitted to a group during one round of categorization are invisible to subsequent feature-set generation and grouping exercises.

Seeding both groups of affixes amounts to an $O(n^2)$ activity. We probe all possible pairs of feature sets for non-normalized, incomplete content overlap, ignoring feature sets of two items or less, which generally belong to spurious bases.[1]. The pair of sets showing the highest overlap

[1]By `incomplete`, we mean we do not consider two feature sets that are identical

Based Affix	Feature Set
ed|0	ed|0 ers|0 ive|0 Re|1 al|0 ory|0 es|0 ion|0 eR|0 able|0 ions|0 ing|0 e|0 s|0 ation|0
ing|0	er|0 ed|0 ive|0 ers|0 Re|1 ory|0 ion|0 es|0 eR|0 able|0 ing|0 e|0 s|0
e|0	ed|0 Re|1 al|0 es|0 eR|0 ion|0 ely|0 e|0 ing|0 s|0 ation|0
t|0	ts|0 ce|0 ting|0 ted|0 tR|0 tion|0 t|0
s|0	ed|0 ers|0 ing|0 Ring|0 s|0 d|0
tion|0	ting|0 ted|0 te|0 t|0 tion|0 tions|0
ers|0	ed|0 er|0 ers|0 ing|0 s|0
ting|0	ts|0 ted|0 ting|0 tion|0 t|0
ation|0	ated|0 ed|0 ations|0 e|0 ation|0
ted|0	ting|0 ted|0 tion|0 t|0 tions|0
d|0	ding|0 rs|0 s|0 ds|0 d|0
ion|0	ed|0 ion|0 ions|0 ing|0 e|0
eR|0	ed|0 eR|0 ing|0 e|0 Re|1
nt|0	nt|0 nce|0 nts|0 ntly|0
on|0	ve|0 ons|0 on|0 ng|0
st|0	st|0 sts|0

Table 4: Sample of First Round of Affix Chaining During the Last Training Iteration of an 39,000 Word English File

initiates the valid group of affixes. The group of invalid affixes initiates as the feature set pair showing the highest internal similarity without demonstrating any coordination with what has become the valid feature set group. That is, at such a point in time, the valid and invalid affix groups are completely distinct in content.

With these two groups in hand, we then begin the process of expanding each through a comparison with each of the remaining unclassified feature sets. We again hunt for content overlap, here, however, normalizing it by the category's total membership. If a feature set's similarity score is greater than .8, it is immediately absorbed; if not, it is set aside until all feature sets have been assessed for classification candidacy. At the end of each such evaluation cycle, we absorb only the highest scoring unclassified feature set as measured against each group.

5.4 Assessment of Membership of Cluster

Next, we massage the membership of the now-filled valid group of affixes, growing it or shrinking it based on degrees of co-occurrence relationships. The idea here is to correct threshold errors pertaining to feature set generation.

To gather such evidence, we examine each affix candidate's productivity, assessing it in terms of both select roots and related affixes. That is, for every affix still in consideration, we tally both the number of co-occurrence associates numbering among the members of the valid affix group and their associated roots. In this, we see how patterns of affixes recommend each other and how fervently they do so. There are two caveats to such an aim. First, we wish to impair invalid affixes from iteratively supporting further invalid affixes. Therefore, in course of tally, we verify that at least one associate belongs to a specially reduced subset of the valid affix group. This portion of members are those classified during the first iteration of the clustering module. The second caveat is one in which we discount any associate that's a sequential subset of a longer group member that could also apply to the shared root. For example, if *d|0* is up for consideration, and its co-occurrence associate is *s|0* via the shared root *stage*, we refrain from increasing *d|0*'s tally of supporting evidential affixes because *es|0* could also characterize a truncated version of the same root candidate.

With this number in hand, establish two separate sets of criteria, one for members of the valid affix group whose current membership we're scrutinzing, another out-of-group membership seeking

admittance.

1. For member of the valid affix group, retain the affix if:
   (a) The number of roots it shares with co-occurrence associates from the valid group exceeds 8% of all the roots it takes
   (b) The affix splits at least 1 root between itself and co-occurrence associates from the valid group AND any of the following
   (c) The number of roots it shares with affix associates from the valid group are are at least of a certain threshold, set to be 2 for files numbering less than 4000 types and .0005% of the types for files larger OR
   (d) Its co-occurrence associates from the reliable subset of the valid group number greater than 33% of its co-occurrence associates from the entire valid group OR
   (e) The number of co-occurrence associates from the reliable subset of the valid group are at least 75% of the total number of affixes composing the reliable subset of the valid group OR
   (f) Its number of shared roots are 75% of the total number of roots it takes
2. For a member outside the valid affix group, admit the affix if:
   (a) The number of its roots demonstrating co-occurrence associates among the valid group is 20% of all the roots it takes AND
   (b) Its co-occurrence associates from the reliable subset of the valid group number greater than 33% of its co-occurrence associates from the entire valid group OR
   (c) Its number of shared roots are 75% of the total number of roots it takes

To strike an appropriate and respectable F-score, we employ one last measure involving previously described modules. We harvest the affixes having survived filtering and pass them as arguments to the affix-chaining subroutine. In doing so, we seek to assess connections or recommendations yielded by what we assume to be a reliable set of affix candidates. Such a strategy marks a return to the hypothesis informing this algorithmic production: valid roots are more productive than invalid roots and the affixes they take cross paths in their distribution. Based upon such conjecture, we therefore expect to see certain repetitions across these new feature-sets/affix chains.

6 Stemming

Once we have a list of presumed affixes in a language, or, at least, a sample thereof, we use it to truncate presumably inflected text. The stemming procedure we employ to this end is quite simple, and is one in which we strip the longest lexical sequence matching an affix. Therefore, for example, if our inducer surmised both *s|0 and es|0* were inflections in English, and the stemmer were presented with *churches*, it would then produce *church* rather than *churche*.

Furthermore, for languages that have the possibility of taking circumfixes, we first stem applicable suffixes, take the resulting foreshortened lexical items, and pass them to the same procedures to evaluate for the presence of prefixes. If a lexical item is found to not have an applicable suffix, the word is transfered as-is to the prefix-stemming component.

Additionally, as a matter of typological practicality, we furthermore place an order of importance on infixation versus traditional word-boundary affixation. As fewer languages demonstrate infixation, we keep a stemmed analysis showing the phenomenon only if the same word does not allow for a match to a traditional affix. Therefore, for example, if both *in|0* and *ing|0* appeared on the list of induced affixes to be stemmed in English, then we would retain *talk + ing|0* over *talkg + in|1*.

Method	Affix Carry-Over Between Iterations	Fourth Affix Chaining
A	No	Yes
B	Yes	Yes
C	No	No

The simplicity of the procedure does yield problems, however. The stemmer has no way of knowing when not to stem candidates that are in fact false matches. For example, if $s|0$ is on the final suffix list for English, and *discuss* is an incoming word, then the former will incorrectly truncate the latter to *discus*, even though the word as passed was itself a complete root. To guard against some mistakes of this ilk, we have instituted as rule that no resulting root can be less than 3 characters. While this does not ameliorate the case of *discuss*, it does amend possibly otherwise frequent missteps. In English, for example, the means guarantees that *is* not reduced to the single character *i*.

Lastly, our stemmer does not yet handle reduplication in the way that our inducer does. In this case, the former program cannot precisely identify purely reduplicated material without the aid of the latter's detection module. Solving this problem as indicated would increase both precision and recall but could severely restrict run-time lexical truncation. This does, however, remain a point for future investigation.

7 Evaluation

7.1 Some Options

For each of the three methods, `A, B` and `C`, we also introduced three further parameters. The first was removing `infixing` from consideration; the second was eliminating the `MLE-based extension` of affix candidates; and the third was allowing suffix training runs to influence prefix selection, which we have termed `co-information`.

File Name	Type Count	Token Count
English2.txt	1,359	4,222
English4.txt	4,783	39,406
English5.txt	3,872	17,067

Table 5: Small English File Names and Type Counts

7.2 The Downside of Productivity

One may notice the clear disparity in results on our home-grown, small files of English newswire and those provided by Morpho Challenge. We attribute this divergence in file size as coupled with poor program execution. The program uses both nested loops to generate feature sets as well as string comparison functions to evaluate the content overlap of those sets. This construction caused no concern on the small files that we were experimenting with, and training could be accomplished in minutes. However, these such alogrithmic features don't lend themselves to linear training time, and we therefore encountered problems when attempting to punctually absorb patterns from even files as large as 200,000 words.

To amend the circumstances, we decided not only to train on a fraction of the data, but to also subdivide that fraction and treat each resulting sub-set of data as an entirely distinct training pass.

Method	File Name	Infixing	MLE Extension	Co-inform	Precision	Recall	Harmonic Mean
Method A	English2.txt	Yes	Yes	No	.83	.48	**.61**
	English4.txt	Yes	No	No	.73	.50	**.59**
		No	No	No	.77	.48	**.59**
Method B	English2.txt	No	No	No	.62	.62	**.62**
	English4.txt	Yes	No	No	.75	.50	**.60**
		No	No	No	.77	.49	**.60**
Method C	English2.txt	Yes	Yes	No	.94	.29	**.44**
	English4.txt	No	No	No	.79	.48	**.60**

Table 6: High Scores for Three Methods Across Three Parameters for Two Small Files of English

Method	MLE	Words Trained	Precision	Recall	Harmonic Mean
Method A	Yes	4,950,000 (5%)	.66	.16	.25.76
Method B	Yes	7,450,000 (9%)	.66	.17	.26
	No	25,750,000 (33%)	.61	.17	.26
Method C	Yes	19,150,000 (25%)	.58	.16	.25
	No	24,900,000 (32%)	.63	.16	.25

Table 7: Scores For Three Methods With Partial Training on the Morpho Challenge 2008 English Corpus

Doing so, we believed, would prevent a burdensome amount of roots and affixes from accruing and therefore limit the cycles of both loops and string-compare functions. Such an assumption was likely born out, as this step managed run-time quite well. It did, however, damage recall more than we expected. The drop in scores, we believe was the exclusive result of data segmentation more so than incomplete examination, as recall remained relatively constant between runs examining 5% of the data to those that examined 6 times as much. Data segmentation is a reliable culprit, in fact, because it ultimately undercut the linguistic productivity represented across the whole corpus. A critical amount of affix connectivity crucial for the development of robust feature set was simply wiped out, in that each run was not informed with the root results of previously processed sub-files. Therefore, a fair amount of valid affixes were not included in the final set of induced sequences given that their distribution was small in scope relative to the training file they found themselves in.

We are currently experimenting with methods to better and more efficiently train on larger files. For one, we are attempting to develop affix feature sets contemporaneously during the trie-based affix and root recovery. This measure involves dual use of both forwards and backwards tries, as, once a root sequence is recovered, it is placed in a reverse trie to guide the recovery of its associated (degree-one) affixes. This process therefore supplants reliance on for-loops nested up to between 3-5 iterative depths. Second, we are weening the code off a preponderance of string-compare functions. We are drafting a new categorization module that stores both the valid and invalid affix groups in separate tries; it therefore tabulates content overlap between each cluster and a candidate feature set by using depth-first search to result in an end-of-affix marker. We are alternately drafting a technique to encode each affix as a number, as comparing integers is quicker than comparing strings. Finally, we are reducing the value of n when attempting to assay the depth feature-set overlap across all pairs. Figuring that sets closer in total membership cardinality show more non-normalized overlap than those sets which are diverse in size, we only compare sets that are the most populous.

Method	Infixes	Words Trained	Precision	Recall	Harmonic Mean
Method A	Yes	350,000 (.6%)	.66	.09	.16
Method B	Yes	750,000 (1%)	.66	.09	.16
Method C	Yes	1,950,000 (3%)	.62	.08	.14
	No	1,950,000 (3%)	.70	.06	.11

Table 8: Scores For Three Methods With Partial Training on the Morpho Challenge 2008 German Corpus

Method	Infixes	Words Trained	Precision	Recall	Harmonic Mean
Method A	Yes	630,000 (1%)	.55	.11	.19
Method B	Yes	630,000 (1%)	.55	.13	.21
Method C	Yes	630,000 (1%)	.56	.12	.20
	No	630,000 (1%)	.56	.11	.18

Table 9: Scores For Three Methods With Partial Training on the Morpho Challenge 2008 Finnish Corpus

References

[1] Delphine Bernhard. Unsupervised morphological segmentation based on segment predictability and word segment alignment. In *PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*, 2006.

[2] Micheal R. Brent, Sreerama K. Murthy, and Andrew Lundberg. Discovering morphemic suffixes: A case study in minimum description length induction. In *Proceedings of the 5^{th} International Workshop on AI and Statistics*, 1995.

[3] Damir Ćavar, Joshua Herring, Toshikazu Ikuta, Paul Rodrigues, and Giancarlo Schrementi. Aligment based induction of morphology grammar and its role for bootstrapping. In Gerhard Ja ger, Poala Monachesi, Gerald Penn, and Shuly Wintner, editors, *Proceedings of Formal Grammar 2004*, pages 47–62.

[4] Sajib Dasgupta and Vincent Ng. Unsupervised word segmentation for bangla. In *Proceedings of the 5^{th} International Conference on Language Processing*. International Conference on Language Processing, 2007b.

[5] Hervé Dejean. Morphemes as necessary concept for structures discovery from untagged corpora. In *Proceedings from the Workshop on Paradigms and Grounding in Natural Language Learning*, pages 295–299, 1998.

[6] John L. Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27:153–198, 2001.

[7] Zelig Harris. From phoneme to morpheme. *Language*, 31:190–222, 1995.

[8] Christian Monson, Jaime Carbonell, Alon Lavie, and Lori Levin. ParaMor: Minimally supervised induction of Paradigm structure and Morphological analysis. In *Proceedings of the 9^{th} Meeting of the ACL Special Interest Group in Computational Morphology and Phonology*, pages 117–125, 2007.