Poor Man's Word-Segmentation: Unsupervised Morphological Analysis for Indonesian

Harald Hammarström
Dept. of Comp. Sci.
Chalmers University
412 96 Gothenburg, Sweden
harald2@chalmers.se

Abstract

We present a partially new fully unsupervised algorithm for morphological segmentation of a arbitrary natural language with only one-slot concatenative morphology. The behaviour of the algorithm is examined in detail for Indonesian as it is a good approximation of such a language. The underlying theory makes no assumptions on whether the language is prefixing or suffixing, or whether affixes are long or short. It does however make the assumption that 1. salient affixes have to be frequent, 2. words essentially are variable length sequences of random characters, and furthermore 3. that a heuristic on what constitutes a systematic affix alteration is valid. The only input required is raw unannotated text and there are no thresholds or parameters that need human tuning. Since there no reliance on existing large data collections or other linguistic resources than raw text, the approach is especially attractive for low-density languages. We will discuss the pressing question whether unsupervised approaches are advantageous over the alternative approach with human-built rules and lexica, especially as it pertains to languages like Indonesian, which do not have much morphology in the first place.

1 Introduction

Indonesian is a language with very little morphology, but nevertheless, there is some. Thus, the first step in the computational treatment of Indonesian is (a tool for) morphological parsing. A traditional approach, be it for Indonesian or another language, is to build a lexicon and morphological rules by hand. We may call this the *manual* approach. In the present paper, we explore another approach, namely to induce a morphological parser from raw (unannotated) text data and, in fact, without any human intervention at all. This will be called the *unsupervised* approach. We will discuss the advantages and disadvantages of both alternatives extensively, with Indonesian as the case at hand.

2 The Problem

The problem at hand can be described as follows:

Input: An unlabeled text corpus of a natural language where

- **(a)** The corpus comes already segmented on the word level.
- **(b)** The language only has one-slot concatenative morphology.

Output: The same text corpus annotated with morphological divisions

Any language with an orthography that shows word-divisions, e.g., with spaces, passes the (a)-restriction right away. So Indonesian, in its contemporary orthography is immediately applicable, while, e.g., Thai is not. The (b)-restriction mandates that the language in question only has concatenative morphology, and furthermore, that affixes cannot be stacked after another, i.e., that there is only maximally one 'slot' for suffixes and maximally one 'slot' for prefixes. In other words, agglutinative languages are not the target of the present problem. Indonesian passes this restriction, or nearly so, as verbal prefixes (like *di-*) cannot be stacked, and the only inflectional suffixes are the personal endings (like *-nya*). On the other hand, multiple affixes do occur somewhat less commonly where the inner suffix is derivational, e.g., *-an-mu* or *-kan-lah*. Similarly, English would be a good one-slot language except for combinations of derivational and inflectional affixes, such as *-ation-s* and latinate prefix stacking such as *dis-en-*.

Considering languages which have concatenative morphology at all, Indonesian is as close to a one-slot language

as one may hope to find [8]. For this reason, Indonesian provides an excellent case for study of unsupervised versus manual approaches to morphological parsing. (The reason for having the one-slot restriction at all, is, that the unsupervised approaches can be considerably less complex.)

3 Manual versus Unsupervised Methods

In the manual approach to morphological parsing, typically, a human implements the following:

(a) A set of morphological rules for parsing and generation.

(b) A lexicon of stems

There are many different frameworks for the implementation of rules and lexicon, e.g., XFST [6] or FM [9] to name a few. However, the choice of framework has little bearing on the issues in this paper, wherefore we disregard this matter further.

The advantages of the manual approach is that one gets a transparent high-accuracy analyser that segments and labels morphologically complex words according to traditional linguistic analysis.

The disadvantages of the manual approach is the human labour required, both for writing the morphological rules and for building the lexicon. On the other hand, in some cases, a finished lexicon-resource is accessible or purchasable as a module, e.g., via the publisher of a dictionary, and then this major part of the human labour is covered already. Another option to reduce the human labour in lexicon-building is to "extract" a lexicon from raw-text. Given morphological paradigms and search constraints, an algorithm extracts a list of stems that are evidenced to belong to a given paradigm in the raw text. A human only needs to skim such a list of extracted lexical items to weed out false positives (see [10, 19, 2, 7] and references therein). Furthermore, any lexicon-based approach to morphological analysis will fail on out-of-lexicon words which inevitably occur in open-domain text (such as newspaper text). As we shall see, there is considerable overlap between out-of-lexicon guessing, lexicon extraction and unsupervised methods.

The unsupervised approach has the advantage that it eliminates the human labour and there is a certain elegance in that the same techniques can be used for different languages.

Naturally, the disadvantage of the unsupervised approaches is that they do not reach full accuracy of morphological analysis. Also, in addition to clear errors, one may expect a certain amount of morpheme segmentations which are slightly at odds with traditional linguistic analysis but not necessarily erroneous. Furthermore, the segmented morphemes are not labeled (or a separate less-than-perfect module is required to learn appropriate labels – see, e.g., [25]) and for many, but not all, applications such labels are in demand.

Various other factors, such as module size or analysis speed, no longer play significant roles, and thus do not contrast between the two alternative approaches. The key issue is how much behind in accuracy unsupervised approaches actually fare – a matter that has so far been rather unclear (see also section 4). One of the main goals of the present paper is to clarify the situation.

A recent survey of published work on morphological analysis for so-called low-density languages shows that the vast majority are of the manual kind [16]. While the contrast between manual and unsupervised approaches are mostly relevant for low-density languages, the division is not so simple. Perhaps as a result of the human labour required, most manually constructed morphological analysers are **not** freely available.

4 Previous Work on Unsupervised Morphological Analysis

A full but concise survey up to late 2007 can be found in [15]. Some relevant work have appeared since, notably [27, 20, 18]. A wide variety of heuristics and models have been employed, which are not of particular importance for this paper, though the interested reader is advised to consult the surveys of [24, 12].

The relevant issue is the fact that nearly all approaches published so far – exceptions include [11, 14] – have a little supervision in the form of thresholds or parameters that have to been set and tuned by a human. This, along with the fact that many approaches target a different (related, but not identical) morphological segmentation problem makes most of this work not directly applicable to the problem considered in the present paper. For these reasons, we will focus on a particular line of work on unsupervised morphological segmentation, which is free from any kind of parameters or thresholds. The (very important) comparison between different unsupervised or little supervised systems is better handled in the annual MorphoChallenge controlled competition[1].

5 Poor Man's Word-Segmentation

In this section we will describe a fully unsupervised algorithm for word-segmentation as applied to Indonesian. The work is an extension of [14]. The following is an outline of the steps and components in the model and its application.

[1]The next edition is the 2009 one `http://www.cis.hut.fi/morphochallenge2009/` accessed 1 May 2009.

Model: A model is compiled from raw text data

Affix Extraction: Extracts a list of affixes ranked according to salience, i.e., how likely they are to be 'real' suffixes in the language in question

Affix Alternation: Given an affix, finds the set of suffixes that systematically appears on the same stems as the given affix

Affix Purging: Given a ranked list of salient affixes (as in the extraction component above), weed out the 'unnecessary' ones, namely those which are covered by a more salient affix

Application: The model is applied to a seen or unseen word and analyses it

True/Random-Ending Heuristic: A word that ends in a salient affix that survived purging is either truly composed of a stem and this affix, or it just happens to end in (or begin with) a character sequence that is identical to a salient affix. The true/random-ending heuristic makes this decision, using affix alternation as a component.

All the above components rely on a number of heuristics which are tailored to the Indonesian language type. The heuristics will be discussed in their due place.

We will illustrate the methods on suffixes, but it is obvious that the same procedures can be used to target prefixes but looking at the word from the opposite direction.

As our input raw text corpus we have used the Indonesian bible [5], solely because of electronic availability, reproduceability and comparability with other languages.

Notation and definitions:

- $w, s, b, x, y, \ldots \in \Sigma^*$: lowercase-letter variables range over strings of some alphabet Σ and are variously called words, segments, strings, etc.
- $s \triangleleft w$: s is a terminal segment of the word w i.e., there exists a (possibly empty) string x such that $w = xs$
- $W, S, \ldots \subseteq \Sigma^*$: capital-letter variables range over sets of words/strings/segments
- $f_W(s) = |\{w \in W | s \triangleleft w\}|$: the (suffix) frequency, i.e., the number of words in W with terminal segment s
- $S_W = \{s | s \triangleleft w \in W\}$: all terminal segments of the wordsin W
- $uf_W(u) = |\{(x, y) | xuy = w \in W\}|$: the substring frequency of u, i.e., the number times u occurs as a substring in the set of words W (x and y may be empty).
- $nf_W(u) = uf_W(u) - f_W(u)$: the non-final frequency of u, i.e. the substring frequency minus those in which it occurs as a suffix.
- $|\cdot|$: is overloaded to denote both the length of a string and the cardinality of a set
- $''$: denotes the empty string

5.1 Affix Extraction

This section briefly describes an Affix Extraction algorithm as applied to Indonesian, see [13, 15] for a fuller description and discussion.

Given a raw text corpus, let W denote the set of words in this corpus. (Note that W is a set, so information about frequency of words in the corpus is disregarded.) Now, assume that $W \subset \{bs | b \in B, s \in S\}$ is actually built up of concatenations of members of two sets of random strings B and S respectively. Assume furthermore that the members of S are frequent, i.e., each s that ends up appearing (as the end of) the words in W occurs with more than random frequency.

The key question is, if words in natural languages are constructed as W above, can we recover the segmentation? That is, can we find B and S, given only W? The answer is yes, we can partially decide this. To be more specific, we can compute a score Z such that $Z(x) > Z(y)$ if $x \in S_W$ and $y \notin S_W$. In general, the converse need not hold, i.e., if both $x, y \in S_W$, or both $x, y \notin S_W$, then it may still be that $Z(x) > Z(y)$. This is equivalent to constructing a ranked list of all possible segments, where the true members of S_W appear at the top, and somewhere down the list the junk, i.e., non-members of S_W, start appearing and fill up the rest of the list. (Thus, it is not said *where* on the list the true-affixes/junk border begins, just that there is a consistent such border. Affix Purging will address the question of junk versus true-affixes.)

Now, how should this list be computed? All terminal segments are contained in the set S_W, the question is just to order them. We shall now define three properties that we argue will be enough to put the S-belonging affixes at the top. For a terminal segment s, define:

Frequency The frequency $f_W(s)$ of s (as a terminal segment).

Curve Drop First, for s, define its curve $C_s(c)$ which is a probability distribution on Σ:

$$C_s(c) = \frac{f_W(cs)}{f_W(s)}$$

Next, more importantly, define its *curve drop* $\overline{C}(s)$ which is a value in $[0, 1]$:

$$\overline{C}(s) = \frac{1 - max_c(C_s(c))}{1 - \frac{1}{|\Sigma|}}$$

Random Adjustment First, for s, define its probability as:

$$P_W(s) = \frac{f_W(s)}{\sum_{s' \in S_W, |s|=|s'|} f_W(s')}$$

Second, equally straightfowardly, for an arbitrary segment u, define its non-final probability as:

$$nP_W(u) = \frac{nf_W(u)}{\sum_{u', |u|=|u'|} nf_W(u')}$$

Finally, for a terminal segment s, define its *random adjustment* $RA(s)$ which a value in $\mathbf{Q}^+$:

$$RA(s) = \begin{cases} \frac{P_W(s)}{nP_W(s)} & \text{if } nP_W(s) > 0 \\ 1.0 & \text{otherwise} \end{cases}$$

It is appropriate now to show the intuition behind the definitions. There isn't much to comment on frequency, so we'll go to curve drop and random adjustment.

The curve drop measure is meant to predict when a suffix is well-segmented to the left. Consider a suffix s, in all the words on which it appears, there is a preceding character c. For example, *-nya* occurs 2035 times, and 672 of those times the preceding character is *a*, 286 times *i*, 227 times *a* and so on, whereas *-ya* occurs 2163 times, 2035 times preceded by *n*, 58 times by *a*, 12 times by *l* and so on. The intuition is as follows. If s is a true suffix and is well-segmented to the left, then its curve-drop value should be high. Frequent true suffixes that attach to bases whose last character is random should have a close to uniform curve. On the other hand, if the curve drop value is low it means there is a character that suspiciously often precedes s. However, if s weren't a true suffix to begin with, perhaps just a frequent but random character, then we expect it's curve drop value to be high too! To exemplify this, we have $\overline{C}(nya) \approx 0.696$, $\overline{C}(ya) \approx 0.061$ and $\overline{C}(s) \approx 0.647$.

The random adjustment measure it precisely to distinguish what a "frequent but random segment" is, that is, discriminate, e.g., *-s* versus *-nya* as well as *-s* versus *-ya*. Now, how does one know whether something is random or not? One approach would be to say the shorter the segment the more random. Although it's possible to get this to work reasonably well in practice, it has some drawbacks. First, it treats all segments of the same length the same, which may be too brutal. Second, it might be too vulnerable to orthography. For example if a language has an odd trigraph for some phoneme, we are clearly going to introduce an error source. Instead we propose that a segment is random iff it has similar probability in any position of the word. This avoids the "flat length"-problems but has others, which we think are less harmful. First, we might get sparse data which can either be back-off smoothed or, like here, effectively ignored (where we lack occurrence we set the RA to 1.0). Second, phonotactic or orthographic constraints may cause curiosities, e.g., English *y* is often spelled *i* when medial as in *fly* vs. *flies*.

To put it all together, we propose the characterization of suffixes in terms of the three properties as shown in table 1. The terms high and low are of course idealized, as they are really gradient properties.

As seen from the table, we hold that true suffixes (and only true suffixes) are those which have a high value for all three properties. Therefore, we define our final ranking score, the $Z_W : S_W \rightarrow \mathbf{Q}$:

$$Z_W(s) = \overline{C}(s) \cdot RA(s) \cdot f_W(s) \quad (1)$$

For the Indonesian we get the top 30 plus bottom 3 suffixes as shown in table 2. The results of this first step do not immediately appear useful, but will be after some post-processing. For example, when a shorter string has higher score than its extension, e.g., *-nya* versus *-tnya*, the longer one should obviously be discarded. It is also worth noting that in cases where Indonesian does show stacked affixes, we get somewhat confused outcomes; sometimes the composition is seen as one simplex segment as in *-anmu* and sometimes the last layer has a higher score than the compostion as in *-lah* versus *-kanlah*.

-anmu	87195.4	*-mu*	6350.9
-nya	73694.0	*-kannya*	5981.8
-anku	51923.8	*-an*	5931.3
-lah	39535.9	*-arlah*	5702.8
-kanlah	27933.2	*-atlah*	5148.6
-hnya	20915.8	*-anlah*	4522.9
-inya	19677.7	*-nglah*	4121.3
-atnya	18361.2	*-anglah*	3990.4
-kan	18318.4	*-akanlah*	3882.9
-tnya	14237.1	*-hlah*	3658.6
-iel	10000.7	*-ah*	3596.1
-snya	9635.2	*-nku*	3359.6
-rlah	9351.8	*-ya*	3203.6
-annya	7802.0	...	0.0
-ilah	7534.3	*-aadil*	0.0
-anya	7005.4	*-aadai*	0.0
-nmu	6531.7	*-aaan*	0.0

Table 2. Top 30 and bottom 3 extracted suffixes for an Indonesian bible corpus.

f_W	$\overline{C}$	RA	Example	Label
high	high	high	*-nya*	True suffix
high	high	low	*-s*	Frequent random segment
high	low	high	*-ya*	Tail of true suffix
high	low	low	N/A	Second part of a digraph
low	high	high	N/A	Infrequent true suffix
low	high	low	*-lempar*	Verb stem with different prefixes
low	low	high	*-akhsaf*	Tail of foreign personal name ending
low	low	low	*-mb*	Infrequent final segment

Table 1. The logically possible configurations of the three suffix properties, accompanied by an appropriate linguistically inspired label and an example from Indonesian.

5.2 Affix Alternation

Given a certain suffix, there is typically a specific set of suffixes that can appear on the same stems, and, except for noise, other suffixes do not may not appear in the same slot. We may call such a set of suffixes that appear on the same stems a *paradigm*. As we shall see, the notion of paradigm is useful for a number of distinctions.

A formalization of same-stem co-occurrence is a measure that takes a set P of suffixes and raw text data, and outputs a number between 0 and 1 according to "how much" the members of P systematically appear on the same stems. It is surprisingly difficult to come up with such a measure as standard vector similarity metrics tend to favour the inclusion of any simply frequent, rather than truly contrasting, terminal segment. Instead, we propose the following usage of co-occurrence statistics.

First, for each suffix x, define its quotient function $H_x(y) : S_W \rightarrow [0, 1]$ as:

$$\frac{|\{s|sx \in W \wedge sy \in W\}|}{|\{s|sx \in W\}|}$$

The formula is conveying the following: We are given a suffix x, and we want to construct a quotient function which is a function from any other suffix to a score between 0 and 1. The score is calculated as: look at all the stems of x, other suffixes y will undoubtedly also occur on some of these stems. For each other suffix y, find the proportion of x:s stems on which y also appears. This proportion will be the quotient associated with y. Two examples of quotient function (sorted on highest value) are given in table 3.

Now, given a set of affixes P, construct a rank by summing the quotient functions of the members of P:

$$V_P(y) = \sum_{x \neq y \in P} H_x(y)$$

The $x \neq y$ is just there so that the y:s that are also in P don't get an "extra" 1.0, since $H_x(x) = 1.0$ regard-

y	$H_{nya}(y)$		y	$H_{mu}(y)$
nya	1.000	mu	1.000	
$''$	0.913	$''$	0.943	
mu	0.261	nya	0.749	
ku	0.153	ku	0.393	
kan	0.071	kan	0.063	
lah	0.057	lah	0.059	
an	0.043	an	0.056	
i	0.039	$kanlah$	0.045	
$kanlah$	0.034	i	0.040	
kah	0.016	kah	0.022	
$ilah$	0.015	$annya$	0.022	
$annya$	0.014	$ilah$	0.019	
$kannya$	0.011	$anmu$	0.015	
t	0.008	n	0.014	
k	0.008	m	0.014	
$anmu$	0.008	k	0.012	
n	0.007	ng	0.011	
m	0.007	$mulah$	0.011	
h	0.007	t	0.009	
s	0.005	$anku$	0.009	
ng	0.005	wan	0.008	
l	0.005	ya	0.007	
$inya$	0.004	ta	0.007	
ya	0.004	s	0.007	
...	...	...	...	...

Table 3. Sample quotient functions/lists for *nya* and *mu*.

less of the data. The rank of y is the number of suffixes s with $V_P(s) < V_P(y)$ – in other words – the place of y on a list of suffixes sorted on on highest V_P. As an example, we can compare in table 4 the very common paradigm $\{-nya, -mu, -'', -ku\}$ with the nonsense paradigm $\{nya, s, a, ya\}$ consisting only of individually frequent suffixes.

y	$V_{P_1}(y)$		y	$V_{P_2}(y)$
”	2.789	”	1.025	
nya	1.652	mu	0.273	
mu	1.004	ku	0.166	
ku	0.572	$snya$	0.098	
lah	0.243	i	0.092	
kan	0.231	kan	0.089	
an	0.197	lah	0.076	
i	0.157	an	0.069	
$kanlah$	0.137	n	0.064	
$annya$	0.075	$anya$	0.058	
$ilah$	0.068	$slah$	0.057	
kah	0.065	$skan$	0.053	
n	0.049	k	0.053	
$anmu$	0.047	ng	0.052	
m	0.043	san	0.048	
t	0.037	t	0.045	
k	0.036	$kanlah$	0.044	
$anku$	0.033	m	0.043	
ng	0.032	si	0.042	
h	0.031	r	0.042	
$mulah$	0.031	h	0.042	
ya	0.031	l	0.030	
ta	0.029	amu	0.030	
s	0.027	**nya**	0.027	
...	...	...	...	...

Table 4. Example ranks for $P_1 = \{-nya, -mu, -'', -ku\}$ and $P_2 = \{nya, s, a, ya\}$.

In table 4, the ranks of the member of P_1 to the left are [0, 1, 2, 3], and for P_2 to the right the ranks are [24, 32, 50, 79].

Now, if we can generalize from these cases it seems that we can rank different hypotheses of paradigms (of the same size) by looking at their quotient ranks. If the members of P “turn up high in” the quotient rank then the members of P tend to turn up on the same stems. There are several issues in formalizing the notion of “turn up high in”. The ranks in the ranked list alone? Also incorporate the scores? Average rank or total sum of ranks? For now we will just do a simple sum of ranks in the ranked list, divide by the optimum sum (which depends on $|P|$ and is $0+\ldots+|P|-1$), and take the inverse. This gives a score between 0 and 1 where a high score means the members of P tend to appear on the same stems:

$$VI(P) = \frac{|P|(|P|-1)}{2\sum_{x\in P} rank(x, V_P)}$$

Note that $VI(P)$ is independent of how well-segmented the members of P are (i.e., how high the Z_W-scores).

The VI-score may be used for a greedy hill-climbing search through the affix set space. For example, we may start with an affix, a one member set, and see whether we can improve the $VI(P)$ score by including another member, and perhaps another after that until we can't improve the score anymore. In this process, we may also entertain the possibility of kicking some member out if that improves the score. Formally, define the growing function of a set P of affixes as:

$$G(P) = argmax_{p\in\{P\}\cap\{P \text{ xor } s|s\in S^W\}} VI(p) \quad (2)$$

$$G^*(P) = \begin{cases} P & \text{if } G(P) = P \\ G^*(G(P)) & \text{if } G(P) \neq P \end{cases} \quad (3)$$

Two growth-examples are shown in table 5, one which attains a perfect 1.0 score and one in which the original member is not a well-segmented to begin with (the pattern such cases exhibits will be exploited for purging).

P	$VI(P)$
(’nya’,)	0.000
(”, ’nya’)	0.333
(”, ’mu’, ’nya’)	0.750
(”, ’ku’, ’mu’, ’nya’)	1.000

P	$VI(P)$
(’s’,)	0.0
(’s’, ’snya’)	0.077
(’s’, ’smu’, ’snya’)	0.273
(’s’, ’sku’, ’smu’, ’snya’)	0.667
(’s’, ’san’, ’sku’, ’smu’, ’snya’)	0.833
(’s’, ’san’, ’skan’, ’sku’, ’smu’, ’snya’)	0.882

Table 5. Example iterations of G^*(’nya’) and G^*(’s’).

5.3 Affix Purging

Now, if we return to the ranked list of suffixes in Z_W as of table 2. As mentioned, one purging heuristic is to discard longer affixes which have a tail whose score is higher. This may be achieved by keeping only those suffixes which are best parse for at least one word.

$$U'_W = \{s|s = argmax_{s'\lhd w} Z_W(s') \text{ for some } w \in W\}$$

This purging heuristic is not sufficient, as a certain amount spurious suffixes – albeit with low Z_W-score – remain, e.g, -*s*. At this point, one could introduce a threshold value to weed out the rest of the spurious suffixes. However, it turns out that there is another heuristic that does the same job, without human intervention.

The behaviour of the G^*('s')-set shown in table 5 is typical for spurious suffixes. Indeed, if s is a spurious suffix, $G(s)$ is likely to consist of s-prefixed to members of a 'true' paradigm. We may therefore posit the following criterion for true-suffixness of a suffix s. Split $G^*(s)$ into $sP = \{x|sx \in G^*(s), |x| > 0\}$, i.e., the members which consist of s followed by some non-empty string versus the rest $\overline{s}P = \{x|x \in G^*(s), x \neq sy, |y| > 0\}$. Note that s itself belongs to $\overline{s}P$ rather than sP and that sP contains the "tail"-strings, stripped of their initial s. If the sum of Z_W-values for $\overline{s}P$ is strictly larger than the the sum of Z_W-values for sP, then the s is a true suffix, otherwise not. Table 6 shows two examples.

The final purged set of suffixes may accordingly be defined as:

$$U''_W = \{s \in U'_W | \sum_{x \in \overline{s}P} Z_W(x) > \sum_{x \in sP} Z_W(x)\}$$

The U''_W for Indonesian boils down to the follwing suffixes:

{anmu, nya, anku, lah, kan, nmu, mu, nku, i, ezer, zabad, inadab, ihud, nadab, arif, obab, ezib, ilene, laf, ilo, ore, e}

The existence of some remaining spurious affixes, such as *-ilene* makes little difference, as these affixes are very infrequent and do not significantly diminish segmentation accuracy. A real error, however, is where Indonesian deviates from being a one-slot language. The lack of *-an*, which is purged out because *-mu* and *-ku* are frequently attached to it.

5.4 True/Random-Ending Heuristic

A word that ends in a salient affix that survived purging is either truly composed of a stem and this affix, or it just happens to end in (or begin with) a character sequence that is identical to a salient affix. The true/random-ending heuristic makes this decision, using affix alternation as a component. Consider the word 'gadisnya' where the *-nya* is a true occurrence of the suffix *-nya*, and the word 'hanya' which just happens to end in the *-nya* character sequence. How can we (heuristically) distinguish the two cases when there is no stem lexicon to tell us that there is a stem *gadis-* but no stem *ha-*?

The intution is the following. If we count all the words in W which begin with *ha-*, they are rather many (247 to be exact). It is not surprising that one of them would just by chance continue as *-nya*, and, crucially, none of the other 246 continuations are *-"*, *-mu* or *-ku*, i.e., the $G^*(nya)$ affixes which systematically alternate with *-nya*. On the other hand, if we look that the words in W which begin with *gadis-*, there are only two more than 'gadisnya', namely 'gadis' and 'gadismu', i.e., with *-"* and *-mu*! To turn this into a formal criterion, given a word $w = xs$ such that $s \in U''_W$, let $C_W(x) = \{y|xy \in W\}$ be the set of "continuations" of x. Extend the notion of final frequency to a set of suffixes P as $f_W(P) = |\{w \in W|w = yz \text{ for some } z \in P\}|$ and let $\alpha_W(P) = f_W(P)/|W|$. The heart of the matter is how much is inside $C_W(x) \cap G^*(s)$ versus how much is outside $C_W(x) \setminus G^*(s)$. Given $w = xs$, if s is just a random continuation of an initial segment x, then many items in $C_W(x) \cap G^*(s)$ will be hard to explain, and if s is truly a well-segmented then too many random continuations in $C_W(x) \setminus G^*(s)$ will overshadow this fact. In other words, the $w = xs$ should be segmented iff the following ratio ≥ 1:

$$\frac{(1 - \alpha_W(G^*(s)))^{|C_W(x) \setminus G^*(s)|}}{\alpha_W(G^*(s))^{|C_W(x) \cap G^*(s)|}}$$

Or equivalently, iff the following ratio ≥ 0:

$$|C_W(x) \setminus G^*(s)| \cdot \log(1 - \alpha_W(G^*(s))) - (|C_W(x) \cap G^*(s)|) \cdot \log \alpha_W(G^*(s))$$

Actual numbers are shown in table 7.

In summary, the following is a procedure for segmenting (or not) a word w (given a training wordset W):

1. Calculate U''_W
2. Consider any $s \triangleleft w$ such that $s \in U''_W$. If there is no such s, then output w (no segmentation)
3. Otherwise, break $w = xs$ and apply the true/random-ending heuristic and output x (segmentation appropriate) or w (no segmentation) accordingly.

5.5 Evaluation

Evaluation was made on a small hand-made test set of 100 word types. 100 word types were selected from W at random and manually segmented both prefix-wise and suffix-wise. For example, 'direncanakannya' was segmented 'di-rencana-kan-nya' and 'mengerutkan' segmented to 'meng-erut-kan' (in all cases where the first character of the root changes as a prefix is added, we arbitrarily chose to define the correct segmentation border so that the result of the mutation belongs to the prefix part). A total of 64 segmentations were found on the 100 words. The unsupervised algorithm was applied to the 100 words, once suffix-wise

s	*-nya*	*-s*
$G^*(s)$	$\{'', ku, mu, nya\}$	$\{s, san, skan, sku, smu, snya\}$
$\overline{s}P$	$\{'', ku, mu, nya\}$	$\{s\}$
sP	$\{\}$	$\{an, kan, ku, mu, nya\}$
$\sum_{x \in \overline{s}P} Z_W(x)$	80790.7	405.5
$\sum_{x \in sP} Z_W(x)$	0.0	105040.5

Table 6. The purging calculation for example suffixes *-nya* (remains) and *-s* (purged out).

w	'gadisnya'	'hanya'
x	*gadis-*	*ha-*
s	*-nya*	*-nya*
$G^*(s)$	$\{'', ku, mu, nya\}$	$\{'', ku, mu, nya\}$
$\alpha_W G^*(s)$	0.205	0.205
$C_W(x)$	$\{'', nya, mu\}$	$\{nya, sratmu, ncurkanlah, rta, rod, \dots\}$
$C_W(x) \cap G^*(s)$	$\{'', nya, mu\}$	$\{nya\}$
$C_W(x) \setminus G^*(s)$	$\{\}$	$\{sratmu, ncurkanlah, rta, rod, \dots\}$
$\lvert C_W(x) \setminus G^*(s)\rvert \cdot \log(1 - \alpha_W(G^*(s)))$	0.0	-56.9
$(\lvert C_W(x) \cap G^*(s)\rvert) \cdot \log \alpha_W(G^*(s))$	-4.8	-1.61

Table 7. The True/Random-Ending Heuristic as applied to the words 'gadisnya' (segmentation appropriate) and 'hanya' (no segmentation).

and once prefix-wise. 58 of segmentations were appropriately found, none spurious and 6 missed (i.e., full precision but 58/64=90.6% recall). All 6 of the missed segmentations were words with stacked affixes such as *-kan-lah*, or with final *-an*.

We know of no purely unsupervised stemming approach to Malay/Indonesian. Published descriptions of work on manual, supervised and semi-supervised Malay/Indonesian stemmers/analyzers include [21, 3, 1, 26, 4, 22, 23, 17][2]. Unfortunately, none of these stemmer/analyzers appears to be available for online processing or as a downloadable program, and, though well-described, require a fair amount of manual labour to reproduce. Therefore we are unable to present accuracy figures for comparison.

6 Discussion

The outlined algorithm is admittedly a complex path of rather untransparent heuristics whose properties we are, at this stage, not able to prove in a mathematically rigourous manner. Nevertheless, all of them have a clear intuition and therefore some of its virtues and errors are easily explainable. Also, as promised, there are no human thresholds or parameters whatsoever.

The accuracy is promising, but we do expect false positives to turn up in larger test sets, such as oversegmentations of *-i* and segmentation with one of the very rare spurious suffixes that survived purging. Undersegmentation, as a result of the one-slot straitjacket, is the main error. Future work, naturally, will focus on extending the heuristics to the multi-slot case. Our feeling is that this is possible, with the same level of accuracy, at the cost of an even more complex web of heuristics.

The present experiment aims to show that, at least for a certain type of languages, unsupervised approaches are competetive accuracywise, and likely to be favoured in a labour/accuracy stand-off. On the other hand, Indonesian morphology is so simple that rule-writing would so take little time, that a hybrid system of hand-written rules and lexicon-less heuristics borrowed from the present approach, would be a serious competitor as well.

7 Conclusion

We have presented a partially novel unsupervised algorithm for morphological segmentation of a arbitrary natural language with only one-slot concatenative morphology. The algorithm achieves high accuracy on Indonesian, a language with little, but mostly one-slot, morphology. An extension of the algorithm to deal systematically with multi-slot morphology is a priority for future work. The presented experiment with Indonesian clarifies the position of unsupervised methods for morphological analysis for low-density languages as an alternative to traditional manual implementation of rules and lexicon.

[2] We are grateful to two anonymous reviewers for suggesting these references.

References

[1] M. T. Abdullah, F. Ahmad, R. Mahmod, and T. M. T. Sembok. Rules frequency order stemmer for malay language. *International Journal of Computer Science and Network Security*, 9(2), 2009.

[2] M. Adler, Y. Goldberg, D. Gabay, and M. Elhadad. Unsupervised lexicon-based resolution of unknown words for full morphological analysis. In *Proceedings of ACL-08: HLT*, pages 728–736, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[3] M. Adriani, J. Asian, B. Nazief, S. M. Tahaghoghi, and H. E. Williams. Stemming indonesian: A confix-stripping approach. *ACM Transactions on Asian Language Information Processing (TALIP)*, 6(4):1–33, 2007.

[4] F. Ahmad, M. Yusoff, and T. M. T. Sembok. Experiments with a stemming algorithm for malay words. *Journal of the American Society for Information Science*, 47(12):909–918, 1996.

[5] American Bible Society. *Alkitab [Indonesian Bible]*. American Bible Society, 2003.

[6] K. R. Beesley and L. Karttunen. *Finite State Morphology*. CSLI Publications, 2003.

[7] C. S. Carlos, M. Choudhury, and S. Dandapat. Large-coverage root lexicon extraction for Hindi. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 121–129, Athens, Greece, March 2009. Association for Computational Linguistics.

[8] M. S. Dryer. Prefixing versus suffixing in inflectional morphology. In B. Comrie, M. S. Dryer, D. Gil, and M. Haspelmath, editors, *World Atlas of Language Structures*, pages 110–113. Oxford University Press, 2005.

[9] M. Forsberg. *Three Tools for Language Processing: BNF Converter, Functional Morphology, and Extract*. PhD thesis, Chalmers University of Technology, Gothenburg, 2007.

[10] M. Forsberg, H. Hammarström, and A. Ranta. Lexicon extraction from raw text data. In T. Salakoski, F. Ginter, S. Pyysalo, and T. Pahikkala, editors, *Advances in Natural Language Processing: Proceedings of the 5th International Conference, FinTAL 2006 Turku, Finland, August 23-25, 2006*, volume 4139 of *Lecture Notes in Computer Science*, pages 488–499. Springer-Verlag, Berlin, 2006.

[11] F. Golcher. Statistical text segmentation with partial structure analysis. In *Proceedings of KONVENS 2006*, pages 44–51. Universität Konstanz, 2006.

[12] J. A. Goldsmith. Segmentation and morphology. In A. Clark, C. Fox, and S. Lappin, editors, *Handbook of Computational Linguistics and Natural Language Processing*. Oxford: Blackwell, [to appear].

[13] H. Hammarström. A naive theory of morphology and an algorithm for extraction. In R. Wicentowski and G. Kondrak, editors, *SIGPHON 2006: Eighth Meeting of the Proceedings of the ACL Special Interest Group on Computational Phonology, 8 June 2006, New York City, USA*, pages 79–88. Association for Computational Linguistics, 2006. `http://www.cs.chalmers.se/~harald2/sigphon06.pdf`.

[14] H. Hammarström. Poor man's stemming: Unsupervised recognition of same-stem words. In H. T. Ng, M.-K. Leong, M.-Y. Kan, and D. Ji, editors, *Information Retrieval Technology: Proceedings of the Third Asia Information retrieval Symposium, AIRS 2006, Singapore, October 2006*, volume 4182 of *Lecture Notes in Computer Science*, pages 323–337. Springer-Verlag, Berlin, 2006.

[15] H. Hammarström. Unsupervised learning of morphology: Survey, model, algorithm and experiments. Thesis for the Degree of Licentiate of Engineering, Department of Computer Science and Engineering, Chalmers University, 91 pp., 2007.

[16] H. Hammarström. A survey of computational morphological resources for low-density languages. To be submitted for Journal of the NEALT, 2009 [in prep.].

[17] L. S. Indradjaja and S. Bressan. Automatic learning of stemming rules for the indonesian language. In *Proceedings of the the 17th Pacific Asia Conference*, pages 62–68, 2003.

[18] M. Johnson. Unsupervised word segmentation for Sesotho using adaptor grammars. In *Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology*, pages 20–27, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[19] K. Lindén. A probabilistic model for guessing base forms of new words by analogy. In A. F. Gelbukh, editor, *Proceedings of CICLing-2008: 9th International Conference on Intelligent Text Processing and Computational Linguistics*, volume 4919 of *Lecture Notes in Computer Science*, pages 106–116. Springer, 2008.

[20] C. Monson. *ParaMor: From paradigm structure to natural language morphology induction*. PhD thesis, Carnegie Mellon University, 2009.

[21] F. Pisceldo, R. Mahendra, R. Manurung, and I. W. Arka. A two-level morphological analyser for the indonesian language. In *Proceedings of the 2008 Australasian Language Technology Association Workshop (ALTA 2008)*, pages 142–150. Hobart, Australia, 2008.

[22] B. Ranaivo. *Reconnaissance automatique de l'affixation en Malais*. PhD thesis, Institut National des Langues et Civilisations Orientales, Paris, France, 2001.

[23] B. Ranaivo-Malançon. Computational analysis of affixed words in malay language. UTMK, USM, Malaysia, 2004.

[24] B. Roark and R. W. Sproat. Machine learning of morphology. In *Computational approaches to morphology and syntax*, volume 4 of *Oxford surveys in syntax and morphology*, pages 116–136. Oxford University Press, 2007.

[25] P. Schone and D. Jurafsky. Language-independent induction of part of speech class labels using only language universals. In *"Machine Learning: Beyond Supervision", Workshop at IJCAI-2001, Seattle, WA, August 2001*, 2001.

[26] S. Y. Tai, C. S. Ong, and N. A. Abullah. On designing an automated malaysian stemmer for the malay language (poster session). In *IRAL '00: Proceedings of the fifth international workshop on on Information retrieval with Asian languages*, pages 207–208, New York, NY, USA, 2000. ACM.

[27] M. Tepper and F. Xia. A hybrid approach to the induction of underlying morphology. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP 2008)*, pages 17–24, Hyderabad, India, January 2008. Asian Federation of Natural Language Processing.