Poor Man's Stemming: Unsupervised Recognition of Same-Stem Words

Harald Hammarström

Chalmers University, 412 96 Gothenburg, Sweden

Abstract. We present a new fully unsupervised human-intervention-free algorithm for stemming for an open class of languages. Since it does not rely on existing large data collections or other linguistic resources than raw text it is especially attractive for low-density languages. The stemming problem is formulated as a decision whether two given words are variants of the same stem and requires that, if so, there is a concatenative relation between the two. The underlying theory makes no assumptions on whether the language uses a lot of morphology or not, whether it 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. Tested on four typologically distant languages, the stemmer shows very promising results in an evaluation against a human-made gold standard.

1 Introduction

The problem at hand can be described as follows:

Input : An unlabeled corpus of an arbitrary natural language and two arbitrary words w_1, w_2 from that language

Output : A YES/NO answer as to whether w_1 and w_2 are morphological variants of one and the same stem (according to traditional linguistic analysis).

Restrictions : We consider only concatenative morphology and assume that the corpus comes already segmented on the word level.

The relevance of the problem is that of stemming as applied in Information Retrieval (IR). The issues of stemming in IR has been discussed at length elsewhere and need not be repeated here. It suffices to say that, though not uncontroversial, stemming continues to be a feature of modern IR systems for languages like English (e.g Google[1]), and is likely to be of crucial importance for languages which make more use of morphology (cf. [1]).

The reasons for attacking the problem in an unsupervised manner include advantages in elegance, economy of time and money (no annotated resources required), and the fact that the same technology may be used on new languages.

[1] According to http://www.google.com/help/basics.html accessed 20 March 2006.

H.T. Ng et al. (Eds.): AIRS 2006, LNCS 4182, pp. 323–337, 2006.

The latter two reasons are especially important in the context of resource-scarce languages.

Our proposed unsupervised same-stem decision algorithm proceeds in two phases. In the first phase, a ranked list of salient affixes are extracted from an unlabeled text corpus of a language. In the second phase, an input word pair is aligned to shortlist affixes that could potentially be added to a common stem to alternate between the two. Crucially, this shortlist of affix alternations is analyzed to check whether they form a *systematic* alternation in the language as a whole (i.e not just in the pair at hand). This analysis depends strongly on the ranked affix list from the first phase.

An outline of the paper is as follows: we start with some notation and basic definitions, with which we describe the theory that is intended to model the assumed behaviour of affixation in natural languages. Then we describe in detail and with examples the thinking behind the affix extraction phase, which actually requires only a few lines to define mathematically. Following that, we present our ideas on how to distinguish a systematic morphological alternation from a spurious one. This part is the more experimental one but at least it requires no guiding, tuning or annotation whatsoever. The algorithm is evaluated against a human gold standard on four languages chosen to span the full width of morphological typology. Finally, we briefly discuss related work, draw some tentative conclusions and hint at future directions.

2 Affix Extraction

We have chosen to illustrate using suffixes but the method readily generalizes to prefixes as well (and even prefixes and suffixes at the same time).

2.1 A Naive Theory of Affixation

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 words in 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

Assume we have two sets of random strings over some alphabet Σ:

- Bases $B = \{b_1, b_2, \ldots, b_m\}$
- Suffixes $S = \{s_1, s_2, \ldots, s_n\}$

Such that:

Arbitrary Character Assumption (ACA): Each character $c \in \Sigma$ should be equally likely in any word-position for any member of B or S.

Note that B and S need not be of the same cardinality and that any string, including the empty string, could end up belonging to both B and S. They need neither to be sampled from the same distribution; pace the requirement, the distributions from which B and S are drawn may differ in how much probability mass is given to strings of different lengths. For instance, it would not be violation if B were drawn from a a distribution favouring strings of length, say, 42 and S from a distribution with a strong bias for short strings.

Next, build a set of affixed words $W \subseteq \{bs | b \in B, s \in S\}$, that is, a large set whose members are concatenations of the form bs for $b \in B, s \in S$, such that:

Frequent Flyer Assumption (FFA): The members of S are frequent. Formally: Given any $s \in S$: $f_W(s) >> f_W(x)$ for all x such that 1. $|x| = |s|$; and 2. not $x \triangleleft s'$ for all $s' \in S$).

In other words, if we call $s \in S$ a *true suffix* and we call x an *arbitrary segment* if it neither a true suffix nor the terminal segment of a true suffix, then any true suffix should have much higher frequency than an arbitrary segment of the same length.

2.2 An Algorithm for Affix Extraction

The key question is, if words in natural languages are constructed as W explained 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_W such that $Z_W(x) > Z_W(y)$ if $x \in S$ and $y \notin S$. In general, the converse need not hold, i.e if both $x, y \in S$, or both $x, y \notin S$, then it may still be that $Z_W(x) > Z_W(y)$. This is equivalent to constructing a ranked list of all possible segments, where the true members of S appear at the top, and somewhere down the list the junk, i.e non-members of S, 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. We shall now define three properties that we argue will be enough to put the S-belonging affixes at the top of the list. For a terminal segment s, define:

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

Curve Drop. The Curve Drop of s is the minimal percentage drop in freqency if s is extended to the left with one character, normalized to the best possible such precentage drop.

$$\overline{C}(s) = \frac{1 - \max_c \frac{f_W(cs)}{f_W(s)}}{1 - \frac{1}{|\Sigma|}} \tag{1}$$

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

$$P_W(s) = \frac{f_W(s)}{\sum_{s' \in S_W} f_W(s')} \tag{2}$$

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

$$nP_W(u) = \frac{nf_W(u)}{\sum_{u'} nf_W(u')} \tag{3}$$

Finally, for a terminal segment s, define its *random adjustment* $RA(s)$ to be the ratio between the two:

$$RA(s) = \begin{cases} \frac{P_W(s)}{nP_W(s)} & \text{if } nP_W(s) > 0 \\ 1.0 & \text{otherwise} \end{cases} \tag{4}$$

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. All examples in this section come from the Brown corpus [2] of one million tokens ($|W| = 47178$ and $|S_W| = 154407$).

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, *-ing* occurs 3258 times, of which it is preceded by t 640 times, of l 329 times, r 317 times, d 258 times, n 249 times and so forth. This contrasts with *-ng* which occurs 3352 times, of which it is preceded by i 3258 times, by o 35 times, by a 26 times and so on. The reasoning is thus 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 $\overline{C}(ing) \approx 0.833$, $\overline{C}(ng) \approx 0.029$ and $\overline{C}(a) \approx 0.851$.

The random adjustment measure it precisely to distinguish what a "frequent but random segment" is, that is, discriminate e.g *-a* versus *-ing* as well as *-a* versus *-ng*. 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, e.g should *-s* be penalized as much as *-a*? Second, it might be considered 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. 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 occurence we set the RA to 1.0). Second, phonotactic or orthographic constraints may cause curiousities, 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.

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

f_W	$\overline{C}$	RA	Example	Label
high	high	high	*-ing*	True suffix
high	high	low	*-a*	Frequent random segment
high	low	high	*-ng*	Tail of true suffix
high	low	low	N/A	Second part of a digraph
low	high	high	*-oholic*	Infrequent true suffix
low	high	low	*-we*	Happenstance low RA-segment?
low	low	high	*-icz*	Tail of foreign personal name ending
low	low	low	*-ebukadnessar*	Infrequent segment

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 $Z_W : S_W \rightarrow \mathbf{Q}$:

$$Z_W(s) = \overline{C}(s) \cdot RA(s) \cdot f_W(s) \tag{5}$$

The final Z_W-score in equation 5 is the one that purports to have the property that $Z_W(x) > Z_W(y)$ if $x \in S_W$ and $y \notin S_W$ – at least if purged (see below). We cannot give a formal proof that languages satisfying ACA and FFA should get a faultless ranking list because this is true only in a heuristic sense. To set bounds on the probability for it to hold is also depends on a lot of factors that are hard, or at least inelegant, to characterize. We hope, however, to have sketched the how the ACA and FFA assumptions are used.

2.3 Affix Extraction Sample Results

On the affix extraction part as such, we will only give some impressionistic results rather than a full-scale evaluation. The reason for this is that, although undoubtedly the list has some valid meaning, it is at present unclear to the author what a gold standard should be in every detail in every language. Furthermore, different applications, such as the final objective in this paper, may not require that a context-less choice between two related affixes, e.g *-ation* and *-tion*, be asserted.

For an English bible corpus [3] we get the top 30 plus bottom 3 suffixes as shown in table 2.

Table 2. Top 30 and bottom 3 extracted suffixes for an English bible corpus. The high placement of English *-eth* and *-iah* are due to the fact that the bible version used has drinketh, sitteth etc and a lot of personal names in *-iah*.

-ed	15448.4	*-ity*	6917.6	*-ts*	3783.1	*-y*	2239.2	*-ded*	1582.2
-eth	12797.1	*-edst*	6844.7	*-ah*	3766.9	*-leth*	2166.3	*-neth*	1540.0
-ted	11899.4	*-ites*	5370.2	*-ness*	3679.3	*-nts*	2122.6	...	...
-iah	11587.5	*-seth*	5081.6	*-s*	3407.3	*-ied*	1941.7	...	...
-ly	10571.2	*-ned*	4826.7	*-ions*	2684.5	*-ened*	1834.9	*-io*	0.0
-ings	8038.9	*-s'*	4305.2	*-est*	2452.6	*-ers*	1819.5	*-ti*	0.0
-ing	7292.8	*-nded*	3833.8	*-sed*	2313.7	*-ered*	1796.7	*-ig*	0.0

The results largely speak for themselves but some comments are in order. A good sign is that the list and its order seems to be largely independent of corpus size (as long as the corpus is not extremely small) but we do get some significant differences between bible English and newspaper English. As is easily seen from the lists, some suffixes are suffixes of each other so one could *purge* the list in some way to get only the most "competitive" suffixes. For a fuller discussion of purging, other languages and all other matters pertaining to the affix extraction algorithm, the reader is referred to the longer exposition in [4].

3 Affix Alternation Analysis

Having a list of salient affixes is not sufficient to parse a given word into stem and affix(es). For example, *sing* happens to end in the most salient suffix yet it is not composed of *s* and *ing* because crucially, there is no **s*, **sed* etc. Thus to parse a given word we have to look at additional evidence beyond the word itself, such as the existence of other inflections of potentially the same stem as the given word, or further, look at inflections of other stems which potentially share an affix with the given word. This line of thought will be pursued below.

The problem at hand, namely, to decide if two given words w_1, w_2 share a common stem (in the linguistic sense) is easier than parsing one word. Essentially, there are four interesting kinds of situations the same-stem-decider must face:

1. w_1 and w_2 do share the same stem and have a salient affix each, e.g *played* vs. *playing*.
2. w_1 and w_2 do share the same stem but one of them has the "zero" affix, e.g *play* vs. *playing*.
3. w_1 and w_2 do not share the same stem (linguistically) but do share some initial segment, e.g *playing* vs. *plough*.
4. w_1 and w_2 do not share the same stem (linguistically) and do not share any initial segment, e.g *playing* vs. *song*.

Number 4 is trivial to decide in the negative. Number 1 is also easy to affirm using a list of salient affixes, whereas the special case of number 2 requires some care. The real difficulty lies in predicting a negative answer for case number 3

(while, of course, at the same time predicting a positive for cases 1 and 2). We will go for an extended discussion of this matter below.

Consider two words $w_1 = xs_1$ and $w_2 = xs_2$ that share some non-empty initial segment x. Except for chance resemblances, which by definition are rare, we would like to say that w_1 and w_2 belong to the same stem iff:

1. s_1 and s_2 are well-segmented salient suffixes in the language, i.e *-w* and *-lt* for *saw* and *salt* are **not**; and
2. s_2 and s_2 must systematically contrast in the language, that is, there must be a large set of stems which can take both s_1 and s_2. For example, the word pair *sting* and *station* align to *-ing* and *-ation* which are both salient suffixes but they do **not** systematically contrast.

The key difficulty is to decide, in an unsupervised manner, when something is systematic and when it isn't. In order to tackle this, we will propose a heuristic for measuring how much two suffixes contrast. This will give a score between 0 and 1 where it is not clear at which value "systematic" begins. We could say that, at this point, the user has to supply a threshold value. However, instead, we devise another heuristic that obviates the need for a threshold at all. The resulting system thus supplies a YES/NO answer to the same-stem decising problem without any human interaction.

3.1 Formalizing Same Stem Co-occurence

From the word distributions characteristic of natural language corpora, it is surprisingly difficult to come up with a measure of how much a set of suffixes show up on the "same stems" that is not such that it favours the inclusion of any simply frequent, rather than truly contrasting, terminal segment. For example, the author has not had much success with standard vector similarity measures. Instead, we propose the following usage of co-occurence statistics. The measure presented is valid for an arbritrary set of suffixes (called P for "paradigm") even though the relevance in this paper is for the case where $|P| = 2$.

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

$$H_x(y) = \frac{|Stems(x) \cap Stems(y)\}|}{|Stems(x)|} \tag{6}$$

where $Stems(x) = \{z|zx \in W\}$. 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 functions (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:

$$V_P(y) = \sum_{x \neq y \in P} H_x(y) \tag{7}$$

Table 3. Sample quotient functions/lists for *ing* and *ed* on the Brown Corpus. H_{ing} and H_{ed} have 68337 and 75853 nonzero values respectively.

y	$H_{ing}(y)$	y	$H_{ed}(y)$
ing	1.00	*ed*	1.00
ed	0.59	*ing*	0.42
″	0.41	″	0.33
s	0.25	*e*	0.21
e	0.24	*s*	0.20
es	0.19	*es*	0.17
er	0.12	*er*	0.08
ers	0.10	*ion*	0.07
ion	0.07	*ers*	0.05
y	0.05	*y*	0.04
ings	0.05	*ions*	0.03
ions	0.03	*ation*	0.03
in	0.03	*able*	0.02
ation	0.03	*ings*	0.02
′s	0.03	*′s*	0.02
ingly	0.03	*or*	0.02
or	0.02	*in*	0.01
able	0.02	*ly*	0.01
ive	0.02	*ive*	0.01
ors	0.02	*ingly*	0.01
ations	0.01	*al*	0.01
er′s	0.01	*ment*	0.01
ment	0.01	*ors*	0.01
ly	0.01	*ations*	0.01
...	...	...	...

Table 4. Example ranks for $P = \{a, an, as, ans, or, orna, ors, ornas\}$ (left) and $P = \{ungen, ig, ar, ts, s, de, ende, er\}$ (right)

y	$VI_P(y)$	y	$VI_P(y)$
a	3.93	”	3.32
an	2.82	t	1.48
or	2.71	a	1.19
”	1.91	r	1.18
orna	1.76	**s**	1.15
ar	1.13	en	1.14
as	1.06	iga	0.86
ade	1.05	d	0.80
ans	0.94	igt	0.73
at	0.89	as	0.66
en	0.82	**de**	0.59
s	0.76	des	0.57
t	0.73	ade	0.55
e	0.71	ung	0.49
er	0.66	**er**	0.49
ad	0.61	at	0.48
ande	0.52	n	0.46
ades	0.47	**ar**	0.45
ats	0.40	an	0.44
i	0.36	e	0.42
...	...	...	...
ors	0.35		
...	...		
ornas	0.27		
...	...		

The $x \neq y$ is just there so that the y:s that are also in P do not get an “extra” 1.0, since $H_x(x) = 1.0$ regardless of the data. The rank is just y sorted on highest $V_P(y)$.

As an example, take W from the Swedish PAROLE-Corpus [5]. We can compare in table 4 the very common paradigm $\{a, an, as, ans, or, orna, ors, ornas\}$ with the nonsense paradigm $\{ungen, ig, ar, ts, s, de, ende, er\}$ consisting only of individually frequent suffixes. In table 4, the ranks of the member of P to the left are [0, 1, 2, 4, 6, 8, 22, 31], and for P to the right the ranks are [115044, 127, 17, 28, 4, 10, 100236, 14].

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 places in the ranked list alone? Also incorporate the scores? Average place or total sum of places? For now we will just do a simple sum of places 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:

$$VI(P) = \frac{|P|(|P|-1)}{2\sum_{x \in P} place(x, V_P)} \tag{8}$$

According to the desiderata 1 and 2 in section 3 (p. 329) we finally define an affix-systematicity likelihood score as:

$$A(P) = VI(P) \sum_{s \in P} Z_W(s) \tag{9}$$

As a convention we set $Z_W('') = 0$.

3.2 Escaping Thresholds

The VI-score from the last section 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 affix 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 – as long as there is no backtracking the search remains polynomial. Formally, define the growing function of a set P of affixes as:

$$G(P) = argmax_{p \in \{P\} \cup \{P \text{ xor } s | s \in S_W\}} VI(p) \tag{10}$$

$$G^*(P) = \begin{cases} P & \text{if } G(P) = P \\ G^*(G(P)) & \text{if } G(P) \neq P \end{cases} \tag{11}$$

Two growth-examples are shown in table 5, one which attains a perfect 1.0 score and one in which the original member is expelled in a later iteration.

Table 5. Example iterations of $G^*('ation')$ and $G^*('xt')$

P	$VI(P)$	P	$VI(P)$
{'ation'}	0.00	{'xt'}	0.00
{'ated', 'ation'}	0.14	{'xt', 'n'}	0.04
{'ate', 'ated', 'ation'}	0.40	{'xt', 'n', 'ns'}	0.12
{'ate', 'ated', 'ating', 'ation'}	0.75	{'n', 'ns'}	0.55
{'ate', 'ated', 'ating', 'ation', 'ations'}	1.00	...	...

Now, how does this help us work around a threshold for deciding how systematically a pair of suffixes have to co-occur to conflate their stems? Recall the writing convention $w_1 = xs_1$ and $w_2 = xs_2$. Instead of having a threshold we may conjecture that:

w_1, w_2 have the same stem iff $s_1 \in G^*(s_2)$ and $s_2 \in G^*(s_1)$

For example, this predicts that *sting* and *station* are not the same stem because neither $G^*(ing) = \{'', e, ed, er, es, ing, s\}$ contains 'ation' nor does $G^*(ation) = \{ate, ated, ating, ation, ations\}$ contain 'ing'. From our experience this test is quite powerful. However, there are of course cases where it predicts wrongly, due to the greedy nature of the G^*-calculation, e.g $G^*(ing)$ does not contain 'ers'. Moreover, if one of the affixes is the empty affix, we need a special fix (see below).

3.3 Same-Stem Decision Algorithm

We can now put all pieces together to define the full algorithm as shown in table 6.

Table 6. Summary of same-stem decision algorithm

Input: A text corpus C and two words w_1, w_2
Step 1. Calculate Z_W as in equation 5
Step 2. Form the set of candidate alignment pairs as:

$$C(w_1, w_2) = \{(s1, s2) | xs_1 = w_1 \text{ and } xs_2 = w_2\} \tag{12}$$

Step 3. If $C(w_1, w_2)$ is empty then answer NO, otherwise pick the best candidate pair as:

$$argmax_{(s1,s2)\in C(w_1,w_2)} A(\{s_1, s_2\}) \tag{13}$$

Step 4. For the winning pair, answer YES/NO acccordingly as $s_1 \in G^*(s_2)$ and $s_2 \in G^*(s_1)$

If one of s_1, s_2 is the empty string then step 3 and 4 should be restated as follows (using s to denote the non-empty one of the two). The maximization value in step 3 should be modified to: $\frac{Z_W(s)}{1+place('', H_s)}$. Step 4 should be modified to: answer YES/NO acccordingly as $'' \in G^*(s)$.

The bad news is that the computation of the G^*:s tends to be slow due to the summing and sorting of typically very long (50 000-ish items) lists. On my standard PC with a Python implementation it typically takes 30 seconds to decide whether two words share the same stem.

4 Evaluation

Several authors, e.g [6,7], have evaluated their stemming algorithms on Information Retrieval performance. While IR is the undoubtedly the major application area we feel that evaluating on retrieval performance does not answer all relevant questions of stemming performance. For instance, a stemmer may make conflations and miss conflations that simply did not affect the test queries. In fact, one may get different best stemmers depending on the test collection. There is also difference as to whether the whole document collection, an abstract of each document or just the query is stemmed.

We find it more instructive to test stemming separately against a stemming gold standard and assess the relevance of stemming for IR by testing the stemming gold standard on IR performance. If stemming turns out to be relevant for IR, then researchers should continue to develop stemming algorithms towards the gold standard. In the other case, one wonder whether IR-improving term conflation methods should be called stemmers.

In order to assess the cross-linguistic applicability of our stemming algorithm we have chosen languages spanning spectrum of morphological typology – from isolating to highly suffixing – Maori, English, Swedish and Kuku Yalanji [8]. As training data we used only the set of words from a bible translation to emphasize the applicability to resource-scarce languages.

For these four languages we devised a stemming gold standard using [12,13] for Maori and [14,15] for Kuku Yalanji, languages not generally known to the author. So as not to let the test set be dominated by too many simple test cases, the selection of test set cases was done as follows:

1. Select a random word w_1 from W for the corresponding language
2. Select a random number i in $0 \leq i \leq |w_1| - 1$
3. Select a random word w_2 from the subset of words from $W \setminus \{w_1\}$ sharing i initial characters with w_1
4. Mark the pair w_1, w_2 to be of the same stem or not, according to traditional linguistic analysis

This was repeated until 200 pairs of words for each language had been selected, 100 same-stem and 100 not same-stem. Except for Maori where we could only really find 13 same-stem cases this way, all involving active-passive alternating verbs (described in detail in [16]).

Table 7. Evaluation results

	Same-stem		Diff.-stem				
Language	Correct	Total	Correct	Total	Language Type	Corpus	Size
Maori	10	13	100	100	Isolating	[9]	NT & OT
English	97	100	100	100	Mildly Suffixing	[3]	NT & OT
Swedish	96	100	100	100	Suffixing	[10]	NT & OT
Kuku Yalanji	94	100	100	100	Strongly Suffixing	[11]	NT & OT Parts

The evaluation results are shown in table 7. Errors fall into just one major type, in which the algorithm is too cautious to conflate; it is when two words do share the stem but where one of the suffixes is rather uncommon (possibly because it is really composite) and therefore it is not in the grow-set of the other suffix; for example Swedish *skap-ade-s* (past passive) and *skap-are-n-s* (agent-noun definite genitive). We also expected false positives in the form of random resemblances involing short words and short affixes; e.g *as* versus *a* but no such cases seem to have occurred in the test set in any of the languages.

We have done attempted a comparison with other existing stemmers, mainly because they tend not be aimed at an open set of languages and those which are,

are really not fully supervised and we fear we might not do justice to them in setting parametres (see Related Work section). The widely known Porter stemmer [17] for English scores exactly the same result for English as our stemmer, which suggests than an unsupervised approach may come very close to explicitly human-informed stemmers. Many other stemmers, however, are superior to ours in the sense that they can stem a single word correctly whereas ours requires a pair of words to make a decision. This is especially relevant when large bodies of data needs to be stem-indexed as it would take quadratic time (in the number of words) in our setting.

5 Related Work

A full survey of stemming algorithms for specific languages or languages like English has more or less fully been done elsewhere (the technology becoming relatively mature cf. [18,19,6,7,20,21,22,23] and references therein). We will focus instead on unsupervised approaches for a wider class of languages.

Melucci and Orio [7] present a very elegant unsupervised stemming model. While training does not require any manually annotated data, some architectural choices depending on the language still has to be supplied by a human. If this can be overcome in an easy way, it would be very interesting to test their Baum-Welch training approach versus the explicit heuristics in this paper, especially on a wider scope of languages than given in their paper. The unsupervised stemmer outlined in [6] actually requires a lot of parametres to be tweaked humanly and mainly targets languages with one-slot morphology.

Other systems for unsupervised learning of morphology which do not explicitly do stemming could easily be transformed into stemmers. Work includes [24,25,26,27,28,29,30,31,32,33,34,35,36,37] and other articles by the same authors. All of these systems, however, require some parametre tweaking as it is and perhaps one more if transformed to stemmers, so there is still work wanting before they can be compared on equal grounds to the stemmer described here. Given that they use essentially the same kind of evidence, it is likely that some of them, especially [38], will reach just as competitive results on the same task.

Of course, we also wish to acknowledge that traditional stemmers output the actual stem, which is one (significant) step further than deciding the same-stem problem for word pairs.

6 Conclusion

We have presented a fully unsupervised human-intervention-free algorithm for stemming for an open class of languages showing very promising accuracy results. Since it does not rely on existing large data collections or other linguistic resources than raw text it is especially attractive for low-density languages. Although polynomial in time, it appears rather slow in practice and may not be suitable for stemming huge text collections. Future directions include investigating whether there

is a speedier shortcut and a better, more systematic, approach to layered morphology i.e for languages which allow affixes to be stacked.

Acknowledgements

The author has benefited much from discussions with Bengt Nordström. We also wish to extend special thanks to ASEDA for granting access to electronic versions of the Kuku Yalanji bible texts.

References

1. Pirkola, A.: Morphological typology of languages for IR. Journal of Documentation **57**(3) (2001) 330–348
2. Francis, N.W., Kucera, H.: Brown corpus. Department of Linguistics, Brown University, Providence, Rhode Island (1964) 1 million words.
3. King James: The Holy Bible, containing the Old and New Testaments and the Apocrypha in the authorized King James version. Thomas Nelson, Nashville, New York (1977)
4. Hammarström, H.: A naive theory of morphology and an algorithm for extraction. In Wicentowski, R., Kondrak, G., eds.: SIGPHON 2006: Eighth Meeting of the Proceedings of the ACL Special Interest Group on Computational Phonology, 8 June 2006, New York City, USA, Association for Computational Linguistics (2006) 79–88
5. Borin, L.: Parole-korpusen vid språkbanken, göteborgs universitet. `http://spraakbanken.gu.se` accessed the 11th of Febuary 2004. (1997) 20 million words.
6. Goldsmith, J., Higgins, D., Soglasnova, S.: Automatic language-specific stemming in information retrieval. In Peters, C., ed.: Cross-Language Information Retrieval and Evaluation: Proceedings of the CLEF 2000 Workshop. Lecture Notes in Computer Science. Springer-Verlag, Berlin (2001) 273–283
7. Melucci, M., Orio, N.: A novel method for stemmer generation based on hidden markov models. In: CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management, New York, NY, USA, ACM Press (2003) 131–138
8. Dryer, M.S.: Prefixing versus suffixing in inflectional morphology. In Comrie, B., Dryer, M.S., Gil, D., Haspelmath, M., eds.: World Atlas of Language Structures. Oxford University Press (2005) 110–113
9. The British & Foreign Bible Society: Maori Bible. The British & Foreign Bible Society, London, England (1996)
10. Svenska Bibelsällskapet: Gamla och Nya testamentet: de kanoniska böckerna. Norstedt, Stockgholm (1917)
11. Summer Institute of Linguistics: Bible: New testament and old testament selctions in kuku-yalanji (1985)
12. Bauer, W., Parker, W., Evans, T.K.: Maori. Descriptive Grammars. Routledge, London & New York (1993)
13. Williams, H.W.: A dictionary of the Maori language. 7 edn. GP Books, Wellington (1971)
14. Patz, E.: A Grammar of the Kuku Yalanji Language of North Queensland. Volume 257 of Pacific Linguistics. Research School of Pacific and Asian Studies, Australian National University, Canberra (2002)

15. Hershberger, H.D., Hershberger, R.: Kuku-Yalanji dictionary. Volume 7 of Work Papers of SIL - AAB. Series B. Summer Institute of Linguistics, Darwin (1982)
16. Sanders, G.: On the analysis and implications of maori verb alternations. Lingua **80** (1990) 149–196
17. Porter, M.F.: An algorithm for suffix stripping. Program **14**(3) (1980) 130–137
18. Erjavec, T., Džeroski, S.: Machine learning of morphosyntactic structure: Lemmatizing slovene words. Applied Artificial Intelligence **18** (2004) 17–41
19. Frakes, W.B., Fox, C.J.: Strength and similarity of affix removal stemming algorithms. SIGIR Forum **37**(1) (2003) 26–30
20. Rogati, M., McCarley, S., Yang, Y.: Unsupervised learning of arabic stemming using a parallel corpus. In: ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, Morristown, NJ, USA, Association for Computational Linguistics (2003) 391–398
21. Hull, D.A.: Stemming algorithms: A case study for detailed evaluation. Journal of the American Soicety for Information Science **47**(1) (1996) 70–84
22. Galambos, L.: Multilingual Stemmer in Web Environment. PhD thesis, Faculty of Mathematics and Physics, Charles University in Prague (2004)
23. Flenner, G.: Ein quantitatives morphsegmentierungssystem für spanische wortformen. In Klenk, U., ed.: Computatio Linguae II: Aufsätze zur algorithmischen und Quantitativen Analyse der Sprache. Volume 83 of Zeitschrift für Dialektologie und Linguistik: Beihefte. Franz Steiner, Stuttgart (1994) 31–62
24. Jacquemin, C.: Guessing morphology from terms and corpora. In: Proceedings, 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '97), Philadelphia, PA. (1997)
25. Yarowsky, D., Wicentowski, R.: Minimally supervised morphological analysis by multimodal alignment. In: Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000). (2000) 207–216
26. Baroni, M., Matiasek, J., Trost, H.: Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In: Proceedings of the Workshop on Morphological and Phonological Learning of ACL/SIGPHON-2002. (2002) 48–57
27. Clark, A.: Learning morphology with pair hidden markov models. In: ACL (Companion Volume). (2001) 55–60
28. Ćavar, D., Herring, J., Ikuta, T., Rodrigues, P., Schrementi, G.: On induction of morphology grammars and its role in bootstrapping. In Jäger, G., Monachesi, P., Penn, G., Wintner, S., eds.: Proceedings of Formal Grammar 2004. (2004) 47–62
29. Brent, M.R., Murthy, S., Lundberg, A.: Discovering morphemic suffixes: A case study in minimum description length induction. In: Fifth International Workshop on Artificial Intelligence and Statistics, Ft. Lauderdale, Florida. (1995)
30. Déjean, H.: Concepts et algorithmes pour la découverte des structures formelles des langues. PhD thesis, Université de Caen Basse Normandie (1998)
31. Snover, M.G., Jarosz, G.E., Brent, M.R.: Unsupervised learning of morphology using a novel directed search algorithm: Taking the first step. In: Workshop on Morphological and Phonological Learning at Association for Computational Linguistics 40th Anniversary Meeting (ACL-02), July 6-12, ACL Publications (2002)
32. Argamon, S., Akiva, N., Amit, A., Kapah, O.: Efficient unsupervised recursive word segmentation using minimum description length. In: COLING-04, 22-29 August 2004, Geneva, Switzerland. (2004)
33. Goldsmith, J.: Unsupervised learning of the morphology of natural language. Computational Linguistics **27**(2) (2001) 153–198

34. Neuvel, S., Fulop, S.A.: Unsupervised learning of morphology without morphemes. In: Workshop on Morphological and Phonological Learning at Association for Computational Linguistics 40th Anniversary Meeting (ACL-02), July 6-12. ACL Publications (2002) 9–15
35. Gaussier, É.: Unsupervised learning of derivational morphology from inflectional lexicons. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-1999), Association for Computational Linguistics, Philadephia (1999)
36. Sharma, U., Kalita, J., Das, R.: Unsupervised learning of morphology for building lexicon for a highly inflectional language. In: Proceedings of the 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON), Philadelphia, July 2002, Association for Computational Linguistics (2002) 1–10
37. Oliver, A.: Adquisició d'informació lèxica i morfosintàctica a partir de corpus sense anotar: aplicació al rus i al croat. PhD thesis, Universitat de Barcelona (2004)
38. Creutz, M., Lagus, K.: Unsupervised models for morpheme segmentation and morphology learning. ACM Transactions on Speech and Language Processing (2006) 1–33