Refining the SED heuristic for morpheme discovery: another look at Swahili

Yu Hu and **Irina Matveeva**
Department of
Computer Science
The University of Chicago
Chicago IL 60637
yuhu@cs.uchicago.edu
matveeva
@uchicago.edu

John Goldsmith
Departments of Linguistics and
Computer Science
The University of Chicago
Chicago IL 60637
ja-goldsmith
@uchicago.edu

Colin Sprague
Department of Linguistics
The University of Chicago
Chicago IL 60637
sprague
@uchicago.edu

Abstract

This paper provides three refinements to the string edit distance-based heuristic for morpheme- and morphology-learning proposed in Anonymous (2005). Experiments with a 7,000 word corpus of Swahili, a language with a rich morphology, support the effectiveness of this approach.

1 Introduction

This paper describes the continuation of work on a technique for the unsupervised learning of the morphology of natural languages which was described in earlier work (Anonymous 2005) and which employs the familiar string edit distance algorithm (Wagner and Fischer 1974 and elsewhere) in its first stage; we refer to it here as the *SED heuristic*. The heuristic described earlier finds 3- and 4-state finite state automata (henceforth, FSAs) from untagged corpora. We continue here our focus on Swahili, a Bantu language of East Africa, because of the very high average number of morphemes per word, especially in the verbal system, a system that presents a real challenge to other systems discussed in the literature.

In Section 2, we present a summary of the work described in Anonymous (2005), presenting the SED heuristic as well as some precision and recall figures for its application to a corpus of Swahili. In Section 3, we propose three elaborations and extensions of this approach, and in Section 4, we describe and evaluate the results from applying these extensions to the corpus of Swahili.[1]

2 SED-based heuristic

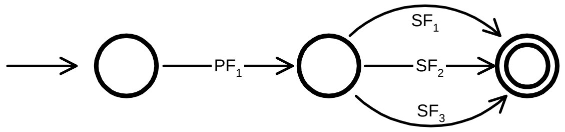
Most systems designed to learn natural language morphology automatically can be viewed as being composed of an initial heuristic component and a subsequent explicit model. The initial or *bootstrapping* heuristic, as the name suggests, is designed to rapidly come up with a set of candidate strings of morphemes, while the model consists of an explicit formulation of either (1) what constitute an adequate morphology for a set of data, or (2) an objective function that must be optimized, given a corpus of data, in order to find the correct morphological analysis.

The best known and most widely used heuristic is due to Zellig Harris (1955) (see also Harris (1967) and Hafer and Weiss (1974) for an evaluation based on an English corpus), using a notion that Harris called successor frequency (henceforth, *SF*). Harris' notion can be succinctly described in contemporary terms: if

[1] SED has been used in unsupervised language learning in a number of studies; see, for example, van Zaanen (2000) and references there, where syntactic structure is studied in a similar context.

we encode all of the data in the data structure known as a trie, with each node in the trie dominating all strings which share a common string prefix,[2] then each branching node in the trie is associated with a morpheme break. For example, a typical corpus of English may contain the words *governed, governing, government, governor,* and *governs*. If this data is encoded in the usual way in a trie, then a single node will exist in the trie which represents the string prefix *govern* and which dominates five leaves corresponding to these five words. Harris's SF-based heuristic algorithm would propose a morpheme boundary after *govern* on this basis. In contemporary terms, we can interpret Harris's heuristic as providing *sets* of simple finite state automata, as in (1), which generate a string prefix (PF_1) followed by a set of string suffixes (SF_i) based on the measurement of a successor frequency greater than 1 (or some threshold value) at the string position following PF_1.

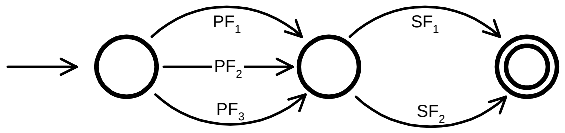
(1)



A variant on the SF-based heuristic, predecessor frequency (henceforth, PF), calls for encoding words in a trie from right to left. In such a PF-trie, each node dominates all strings that share a common string suffix. In general, we expect SF to work best in a suffixing language, and PF to work best in prefixing language; Swahili, like all the Bantu languages, is primarily a prefixing language, but it has a significant number of important suffixes in both the verbal and the nominal systems.

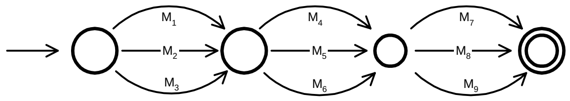
Goldsmith (2001) argues for using the discovery of signatures as the bootstrapping heuristic, where a *signature* is a maximal set of stems and suffixes with the property that all combinations of stems and suffixes are found in the corpus in question. We interpret Goldsmith's signatures as extensions of FSAs as in (1) to FSAs as in (2); (2) characterizes Goldsmith's notion of signature in term of FSAs. In particular, a signature is a set of forms that can be characterized by an FSA of 3 states.

(2)



Anonymous (2005) proposed a simple alternative heuristic which utilizes the familiar dynamic programming algorithm for calculating string-edit distance, and finding the best alignment between two arbitrary strings (Wagner and Fischer 1974). The algorithm finds subsets of the data that can be exactly-generated by sequential finite state automata of 3 and 4 states, as in (3), where the labels m_i should be understood as cover terms for morphemes in general. An automaton *exactly-generates* a set of strings S if it generates all strings in S and no other strings; a *sequential* FSA is one of the form sketched graphically in (1)-(3), where there is a unique successor to each state.

(3)

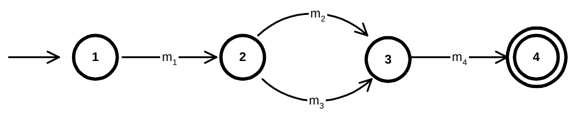


2.1 First stage: alignments.

If presented with the pair of strings *anapenda* and *anamupenda* from an unknown language, it is not difficult for a human being to come up with the hypothesis that *mu* is a morpheme inside a larger word that is composed of at least two morphemes, perhaps *ana-* and *-penda*. The SED heuristic makes this observation explicit by building small FSAs of the form in (4), where at most one of m_1 or m_4 may be null, and at most one of m_2 and m_3 may be null: we refer to these as *elementary alignments*. The strings m_2 and m_3 are called *counterparts*; the pairs of strings m_1 and m_4 are called the *context* (of the counterparts).

[2] We use the terms *string prefix* and *string suffix* in the computer science sense: a string S is a string prefix of a string X iff there exists a string T such that 0058 = S.T, where "." is the string concatenation operator; under such conditions, T is likewise a *string suffix* of X. Otherwise, we use the terms *prefix* and *suffix* in the linguistic sense, and a string prefix (e.g., *jump*) may be a linguistic stem, as in *jump-ing*.

(4)

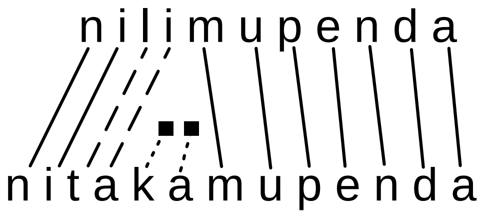


The first stage of the algorithm consists of looking at all pairs of words S, T in the corpus, and passing through the following steps:

We apply several initial heuristics to eliminate a large proportion of the pairs of strings before applying the familiar SED algorithm to them, in view of the relative slowness of the SED algorithm; see Anonymous (2005) for details.

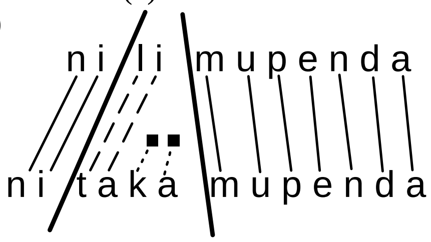
We compute the optimal alignment of S and T using the SED algorithm, where alignment between two identical letters (which we call *twins*) is assigned a cost of 0, alignment between two different letters (which we call *siblings*) is assigned a cost of 1.5, and a letter in one string not aligned with a segment on the other string (which we call an *orphan*) is assigned a cost of 1. An alignment as in (5) is thus assigned a cost of 5, based on a cost of 1 assigned to each broken line, and 1.5 to each dotted line that ends in a square box.

(5)



There is a natural map from every alignment to a unique sequence of pairs, where every pair is either of the form (S[i], T[j]) (representing either a twin or sibling case) or of the form (S[i], 0) or (0, T[j]) (representing the orphan case). We then divide the alignment up into perfect and imperfect spans: perfect spans are composed of maximal sequences of twin pairs, while imperfect spans are composed of maximal sequences of sibling or orphan pairs. This is illustrated in (6).

(6)



There is a natural equivalence between alignments and sequential FSAs as in (4), where perfect spans correspond to pairs of adjacent states with unique transitions and imperfect spans correspond to pairs of adjacent states with two transitions, and we will henceforth use the FSA notation to describe our algorithm.

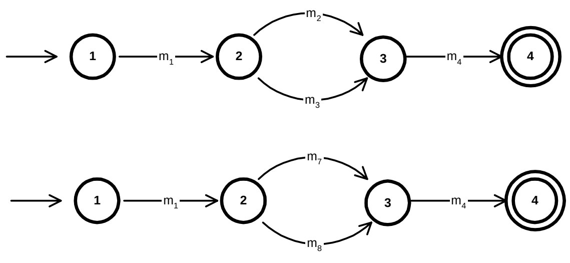
2.2 Collapsing alignments

As we noted above (4), for any elementary alignment, a *context* is defined: the pair of strings (one of them possibly null) which surround the pair of *counterparts*. Our first goal is to collapse alignments that share their context. We do this in the following way.

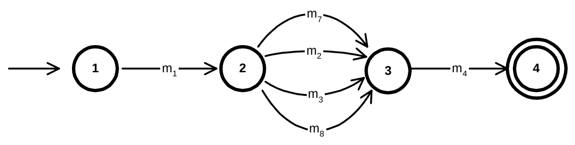
Let us define the set of strings associated with the paths leaving a state S as the *production* of state S. A four-state sequential FSA, as in (4), has three states with non-null productions; if this particular FSA corresponds to an *elementary alignment*, then two of the state-productions contain exactly one string—and these state-productions define the *context*— and one of the state-productions contains exactly two strings (one possibly the null string)—this defines the *counterparts*. If we have two such four-state FSAs whose context are identical, then we collapse the two FSAs into a single *conflated* FSA in which the context states and their productions are identical, and the states that produced the *counterparts* are collapsed by creating a state that produces the union of the productions of the original states. This is illustrated in (7): the two FSAs in (7a) share a context, generated by their states 1 and 3, and they are collapsed to form the FSA in (7b), in which the context states remain unchanged, and the counterpart states, labeled ‘2’, are collapsed to form a new state ‘2’ whose production is the union of the productions of the original states.

(7)

a.



b.



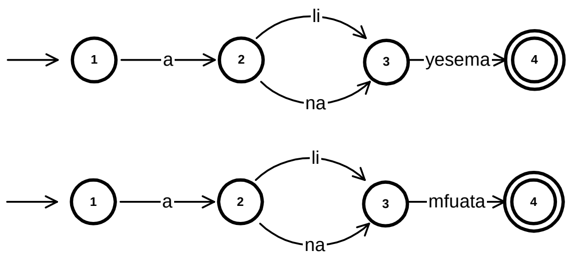
2.3 Collapsing the resulting sequential FSAs

We now generalize the procedure described in the preceding section to collapse any two sequential FSAs for which all but one of the corresponding states have exactly the same production. For example, the two sequential FSAs in (8a) are collapsed into (8b).

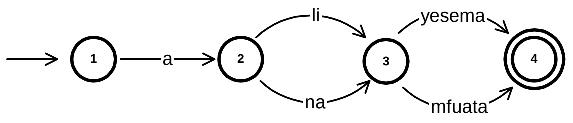
Three and four-state sequential FSAs as in (8b), where at least two of the state-transitions generate more than one morpheme, form the set of *templates* derived from our bootstrapping heuristic. Each such *template* can be usefully assigned a quantitative score based on the number of letters "saved" by the use of the template to generate the words, in the following sense. The template in (8b) summarizes four words: *aliyesema, alimfuata, anayesema,* and *anamfuata*. The total string length of these words is 36, while the total number of letters in the strings associated with the transitions in the FSA is 1+4+12 = 17; we say that the FSA *saves* 36-17 = 19 letters. In actual practice, the significant templates discovered save on the order of 200 to 5,000 letters, and ranking them by the number of letters saved is a good measure of how significant they are in the overall morphology of the language. We refer to this score as a template's *robustness*; we employ this quantity again in section 3.1 below.

By this ranking, the top template found in our Swahili corpus of 50,000 running words was one that generated *a* and *wa* (class 1 and 2 subject markers) and followed by 246 correct verb continuations (all of them polymorphemic); the first 6 templates are summarized informally in Table 1. We note that the third and fourth template can also be collapsed to form a template of the form in (3), a point we return to below. Precision, recall, and F-score for these experiments are given in Table 2. The gold-standard used in the current paper is somewhat larger than that used in the previous paper, but the changes are not great, and the effect on the numbers is small.

(8)
a.



b.



State 1	State 2	State 3
a, wa (sg., pl. human subject markers)	246 stems	
ku, hu (infinitive, habitual markers)	51 stems	
wa (pl. subject marker)	*ka, li* (tense markers)	25 stems
a (sg. subject marker)	*ka, li* (tense markers)	29 stems
a (sg. subject marker)	*ka, na* (tense markers	28 stems
37 strings	*w* (passive marker)	*a*

Table 1 Top templates in Swahili

	Precision	Recall	F-score
SED	0.77	0.57	0.65
SF	0.54	0.14	0.22
PF	0.68	0.20	0.31

Table 2 Previously reported results

3 Proposed developments

In this section, we describe three developments of the SED-based heuristic sketched in section 2. The first disambiguates which state it is that string material should be associated with in cases of ambiguity; the second *collapses* templates associated with similar morphological structure; the third uses the FSAs to *predict* words that do not actually occur in the corpus by hypothesizing stems on the basis of the established FSAs and as yet unanalyzed words in the corpus.

3.1 Disambiguating FSAs

In the case of a sequential FSA, when the final letter of the production of a (non-final) state S are identical, then that letter can be moved from being the string-suffix of all of the productions of state S to being the string-prefixes of all of the productions of the following state. More generally, when the *n* final letters of the productions of a state are identical, there is an n-way ambiguity in the analysis, and the same holds symmetrically for the ambiguity that arises when the *n* initial letters of the production of a (non-initial) state.

Thus two successive states, S and T, must (so to speak) fight over which will be responsible for generating the ambiguous string. We employ two steps to disambiguate these cases.

Step 1: The first step is applicable when the *number* of distinct strings associated with states S and T are quite different in size (typically corresponding to the case where one generates *grammatical* morphemes and the other generates *stems*); in this case, we assign the ambiguous material to the state that generates the smaller number of strings. There is a natural motivation for this choice from the perspective of our desire to minimize the size of the grammar, if we consider the size of the grammar to be based, in part, on the sum of the lengths of the morphemes produced by each state.

Step 2: It often happens that an ambiguity arises with regard to a string of one or more letters that could potentially be produced by either of a pair of successive states involving grammatical morphemes. To deal with this case, we make a decision that is also (like the preceding step) motivated by a desire to minimize the description length of the grammar. In this case, however, we think of the FSA as containing explicit *strings* (as we have assumed so far), but rather *pointers* to strings, and the "length" of a pointer to a string is inversely proportional to the logarithm of its frequency. Thus the *overall* use of a string in the grammar plays a crucial role in determining the length of a grammar, and we wish to maximize the appearance in our grammar of morphemes that are used frequently, and minimize the use of morphemes that are used rarely.

We implement this idea by collecting a table of all of the morphemes produced by our FSA, and assigning each a score which consists of the sum of the robustness scores of each template they occur in (see discussion just above (8)). Thus morphemes occurring in several high robustness templates will have high scores; morphemes appearing in a small number of lowly ranked templates will have low scores.

To disambiguate strings which could be produced by either of two successive states, we consider all possible parsings of the string between the states, and score each parsing as the sum of the scores of the component morphemes; we chose the parsing for which the total score is a maximum.

For example, Swahili has two common tense markers, *ka* and *ki*, and this step corrected a template from {*ak*}+{*a*,*i*}+{stems} to {*a*}+{*ka*,*ki*}+{stems}, and others of similar form. It also did some useful splitting of joined morphemes, as when it modified a template {*wali*} + {NULL, *po*} + {stems} to {*wa*} + {*li*, *lipo*} + {stems}. In this case, *wali* should indeed be split into *wa* + *li* (subject and tense markers, resp.), and while the change creates an error (in the sense that *lipo* is, in fact, two morphemes; *po* is a subordinate clause marker), the resulting error occurs considerably less often in the data, and the correct template will better be able to be integrated with out templates.

3.2 Template collapsing

From a linguistic point of view, the SED-based heuristic creates too many FSAs because it stays too close to the data provided by the corpus. The only way to get a more correct grammar is by collapsing the FSAs, which will have as a consequence the generation of new words not found in the corpus. We apply the following relatively conservative strategy for collapsing two templates.

We compare templates of the same number of states, and distinguish between states that produce grammatical morphemes (five or fewer in number) and those that produce stems (that is, lexical morphemes, identified as being six or more in number). We collapse two templates if the productions of the corresponding states satisfy the following conditions: if the states generate stems, then the intersection of the productions must be at least two stems, while if the states are grammatical morphemes, then the productions of one pair of corresponding states must be *identical*, while for the other pair, the symmetric difference of the productions must be no greater than two in number (that is, the number of morphemes produced by the state of one template but not the other must not exceed 2).

3.3 Reparsing words in the corpus and predicting new words

When we create robust FSAs—that is, FSAs that generate a large number of words—we are in a position to go back to the corpus and reanalyze a large number of words that could not be previously analyzed. That is, a 4-state FSA in which each state produced two strings generates 8 words, and *all* 8 words must appear in the corpus for the method described so far in order for this particular FSA to generate *any* of them. But that condition is unlikely to be satisfied for any but the most common of morphemes, so we need to go back to the corpus and *infer* the existence of new stems (as defined operationally in the preceding paragraph) based on their occurrence in several, but not *all* possible, words. If there exist 3 distinct words in the corpus which would all be generated by a template if a given stem were added to the template, we add that stem to the template.

4 Experiments and Results

In this section, we present three sets of evaluations of the refinements of the SED heuristics described in the preceding section. We used a corpus of 7,180 distinct words occurring in 50,000 running words from a Swahili translation of the Bible obtained on the internet.

4.1 Disambiguating FSAs

In order to evaluate the effects of the disambiguating of FSAs described in section 3.1, we compare precision and recall of the identification of morpheme boundaries using the SED method *with* and *without* the disambiguation procedure described above. In Figures 1 and 2, we graph precision and recall for the top 10% of the templates, displayed as the leftmost point, for the top 20% of the templates, displayed as the second point from the left; and so on. We see that the disambiguation repairs almost 50% of the previous errors, and increases recalls by about 10%. With these increases in precision and recall, it is clear that the disambiguating step provides a considerably more accurate morpheme boundary discovery procedure.

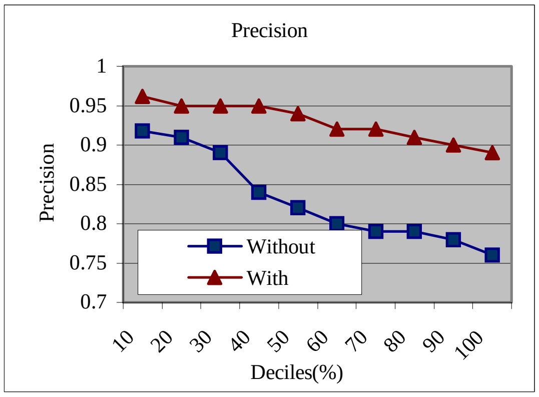


Figure 1 Comparison of precision

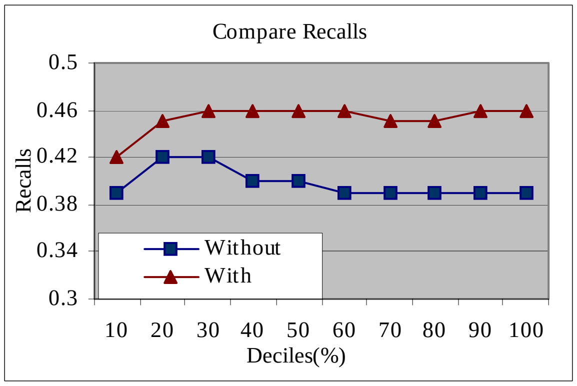


Figure 2 Comparison of recall

4.2 Template collapsing

The second refinement discussed above consists of finding pairs of similar templates, collapsing them as appropriate, and thus creating patterns that generate new words that did not participate in the formation of the original templates. These new words may or may not themselves appear in the corpus. We are, however, able to judge their morphological well-formedness by inspection. We list in Table 3 the entire list of eight templates that are collapsed in this step.

All of the templates which are collapsed in this step are in fact of the same morphological structure (with one very minor exception[3]): they are of the form *subject marker* + *tense marker* + *stem*, and the collapsing induced in this procedure correctly creates larger templates of precisely the same structure, generating new words not seen in the corpus that are in fact correct.

4.3 Reparsing

After previous refinements, we obtain a number of robust FSAs, for example, those collapsed templates in Table 3. With them, we then search the corpus for those words that can only be partly fitted into these FSAs and generate associated stems. Table 4 shows the reparsed words that had not been parsed by earlier templates and also newly added stems for some robust FSAs (the four collapsed templates in Table 3). Stems such as *anza* 'begin' and *fanya* 'do' are thus added to the first template, and all words derived by prepending a tense marker and a subject marker are indeed accurate words. As the words in Table 4 suggest, the reparsing process adds new, common stems to the stem-column of the templates, thus making it easier for the collapsing function to find similarities across related templates.

In future work, we will take use the larger templates, populated with more stems, and input them to the collapsing function described in 3.2.

5 Conclusions

On the basis of the experiments with Swahili described in this paper, the additions to the SED heuristic discussed in this paper appear to be useful tools for the discovery of morphemes in languages with rich morphologies, and for the discovery of the FSAs that constitute the morphologies of those languages.

[3] The exception involves the distinct morpheme *po*, a subordinate clause marker which must ultimately be analyzed as appearing in a distinct template column to the right of the tense markers.

	One Template	The other template	Collapsed Template	Example of a created word
1	{a}-{ka,na}-{stems}	{a}-{ka,ki}-{stems}	{a}-{ka,ki,na}-{stems}	a-ki-mwona
2	{wa}-{ka,na}-{stems}	{wa}-{ka,ki}-{stems}	{wa}-{ka,ki,na}-{stems}	wa-na-msifu
3	{a}-{ka,ki,na}-{stems}	{wa}-{ka,ki,na}-{stems}	{a,wa}-{ka,ki,na}-{stems}	wa-na-jua
4	{a}-{liye,me}-{stems}	{a}-{liye,li}-{stems}	{a}-{liye,li,me}-{stems}	a-li-mwona
5	{a}-{ki,li}-{stems}	{wa}-{ki,li}-{stems}	{a,wa}-{ki,li}-{stems}	a-ki-kataa
6	{a}-{lipo,li}-{stems}	{wa}-{lipo,li}-{stems}	{a,wa}-{lipo,li}-{stems}	wa-lipo-kuja
7	{a,wa}-{ki,li}-{stems}	{a,wa}-{lipo,li}-{stems}	{a,wa}-{ki,lipo,li}-{stems}	a-lipo-shuka
8	{a}-{na,naye}-{stems}	{a}-{na,ta}-{stems}	{a}-{na,ta,naye}-{stems}	a-naye-mpenda

Table 3 Collapsed Templates and Created Words Sample.

	Template	Reparsed Words Not Parsed Before	Added Stems
1	{a, wa}-{ka,ki,na}-{stems}	akawakweza, akiwa, anacho, akibatiza, …	toka, anza, waita, fanya, enda, …
2	{a}-{li,liye,me }-{stems}	ameinuka, ameugua, alivyo, aliyoniagiza, …	zaliwa, kuwa, fanya, sema
3	{a, wa}-{ki,li,lipo}-{stems}	alimtoboa, alimtaka, waliamini, …	pata, kuwa, kaa, fanya, chukua, fika, …
4	{a} – {na,naye,ta}-{stems}	analazwa, atanitukuza, anaye, anakuita, …	ingia, sema

Table 4 Reparsed words and "discovered" stems

References

Goldsmith, John. (2001). Unsupervised Learning of the Morphology of a Natural Language. *Computational Linguistics* 27(2): 153-198.

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

Harris, Zellig. (1955). From Phoneme to Morpheme. *Language* 31: 190-222.

Harris, Zellig. (1967). Morpheme Boundaries within Words: Report on a Computer Test. *Transformations and Discourse Analysis Papers* 73.

Wagner, R. A., Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the Association for Computing Machinery* 21(1): 168-73.

van Zaanen, Menno. 2000. ABL: Alignment-Based Learning. *Proceedings of the 17th Conference on Computational Linguistics*, vol. 2. p. 961-67.