Unsupervised learning of derivational morphology from inflectional lexicons

Éric Gaussier
Xerox Research Centre Europe 6, Chemin de Maupertuis 38240 Meylan F.
Eric.Gaussier@xrce.xerox.com

Abstract

We present in this paper an unsupervised method to learn suffixes and suffixation operations from an inflectional lexicon of a language. The elements acquired with our method are used to build stemming procedures and can assist lexicographers in the development of new lexical resources.

1 Introduction

Development of electronic morphological resources has undergone several decades of research. The first morphological analyzers focussed on inflectional processes (inflection, for English, mainly covers verb conjugation, and number and gender variations). With the development of Information Retrieval, people have looked for ways to build simple analyzers which are able to recognize the stem of a given word (thus addressing both inflection and derivation[1]). These analyzers are known as stemmers.

Faced with the increasing demand for natural language processing tools for a variety of languages, people have searched for procedures to (semi-)automatically acquire morphological resources. On the one hand, we find work from the IR community aimed at building robust stemmers without much attention given to the morphological processes of a language. Most of this work relies on a list of affixes, usually built by the system developer, and a set of rules to stem words (Lovins, 1968; Porter, 1980). Some of these works fit within an unsupervised setting, (Hafer and Weiss, 1974; Adamson and Boreham, 1974) and to a certain extent (Jacquemin and Tzoukerman, 1997), but do not directly address the problem of learning morphological processes. On the other hand, some researchers from the computational linguistics community have developed techniques to learn affixes of a language and software to segment words according to the identified elements. The work described in (Daelemans et al., 1999) is a good example of this trend, based on a supervised learning approach. However, it is difficult in most of these studies to infer the underlying linguistic framework assumed.

[1]The distinction between inflectional and derivational morphology is far from clearcut. However, in practice, such a distinction allows one to divide the problems at hand and was implicitly adopted in our lexicon development plan.

We present in this paper an unsupervised method to learn suffixation operations of a language from an inflectional lexicon. This method also leads to the development of a stemming procedure for the language under consideration. Section 2 presents the linguistic view we adopt on derivation. Section 3 describes the preliminary steps of our learning method and constitutes the core of our stemming procedure. Finally, section 4 describes the learning of suffixation operations.

2 Derivation in a language

The derivational processes of a language allow speakers of that language to analyse and generate new words. Most recent linguistic theories view these processes as operations defined on words to produce words. From a linguistic point of view, a word can be represented as an element made of several independent layers (feature structures, for example, could be chosen for this representation. We do not want to focus on a particular formalism here, but rather to explain the model we will adopt). The different layers and the information they contain vary from one author to the other. We adopt here the layers used in (Fradin, 1994), as exemplified on the French noun *table*:

(G)	table
(F)	(teibl)
(M)	fem-sg
(SX)	N
(S)	**table**

where (G) corresponds to the graphemic form of the word, (F) to the phonological form, and (M), (SX) and (S) respectively contain morphological, syntactic and semantic information. A derivation process then operates on such a structure to produce a new structure. Each layer of the original structure is transformed via this operation.

We can adopt the following probabilistic model to account for such a derivation process:

$$P(w_1 \rightarrow w_2) = \sum_{Op} p(w_2(G) = Op_G(w_1(G)),$$
$$w_2(F) = Op_F(w_1(F)), w_2(M) = Op_M(w_1(M)),$$
$$w_2(SX) = Op_{SX}(w_1(SX)), w_2(S) = Op_S(w_1(S)))$$

where Op is a derivation process, Op_G is the component of Op which operates on the graphemic layer, and $w(G)$ is the graphemic layer associated to word w.

The different layers can be divided up into three main dimensions, used in linguistic studies to identify and classify suffixes of a language: the formal dimension (corresponding to G and F), the morphosyntactic dimension (M and SX), and the semantic dimension (S). The nature of the operation along these dimensions mainly depend on the language under consideration. For example, for Indo-European languages, for the formal dimension, a suffixation operation consists in the concatenation of a suffix to the original form. Morphographemic as well as phonological rules are then applied to turn the ideal form obtained via concatenation into a valid surface form.

We focus in this article on concatenative languages, *id est* languages for which derivation corresponds, for the formal dimension, to a concatenation operation. We also restrict ourselves to the study of suffixes and suffixation. Nevertheless, the principles and methods we use can be extended to non-concatenative languages and prefixes.

The aim of the current work is two-fold. On the one hand we want to develop stemming procedures for Information Retrieval. On the other hand, we want to develop methods to assist lexicographers in the development of derivational lexicons. We posses inflectional lexicons for a variety of different languages, and we'll use these lexicons as input to our system. Furthermore, we are interested in making the method as language independent as possible, which means that we will explore languages without a priori knowledge[2] and thus we wish to rely on an unsupervised learning framework.

The probabilistic model above suggests that, in order to learn suffixation operations of a language, one should look at word pairs (w_1, w_2) of that language for which w_2 derives from w_1. In an unsupervised setting, such pairs are not directly accessible, and we need to find ways to extract the information we are interested in from a set of pairs the words of which are not always related via derivation. The method we designed first builds, for a given language, relational families, which are an approximation of derivational families. These families are then used to produce pairs of words which are a first approximation of the pairs of related words. From this set of pairs, we then extract suffixes and suffixation operations.

[2] In particular, it is interesting to avoid relying on suffix lists, which vary from one author to the other.

The next section addresses the construction of relational families.

3 Construction of relational families

Our goal is to build families which are close to the derivational families of a language. This contruction relies on the notion of suffix pairs that we explain below.

3.1 Extraction of suffix pairs

The intuition behind the extraction of suffixes is that long words of a given language tend to be obtained through derivation, and more precisely through suffixation, and thus could be used to identify regular suffixes.
We first define a measure of similarity between words based on the comparison of truncations.
Definition 1: *two words w_1 and w_2 of a given language L are said to be* **p-similar** *if and only if*

i $trunc(w_1, p) \equiv trunc(w_2, p)$, where $trunc(w, k)$ is composed of the first k characters of w,

ii there is no q such that: $q > p$ and $trunc(w_1, q) \equiv trunc(w_2, q)$

The equivalence relation ($\equiv$) defined on the alphabet of L allows one to capture orthographic variants, such as the alternation *c* - *ç* in French. The character strings s1 and s2 obtained after truncation of the first p characters from two p-similar words are called pseudo-suffixes. The pair (s1,s2) will be called a pseudo-suffix pair of the language L, and will be said to link w_1 and w_2. Note that the strings s1 and/or s2 may be empty: they are both empty if the words w_1 and w_2 differ only in their part of speech, in which case we speak about conversion.
The above definition allows us to state that the English words "deplorable" and "deploringly" are 6-similar and that (able,ingly) is an English pseudo-suffix pair. Since "deplorable" is an adjective and "deploringly" is an adverb, we can provide a more precise form for the pseudo-suffix pair, and write (able+AJ,ingly+AV) (where +AJ stands for adjective and +AV for adverb) with the following interpretation: we can go from an adjective (resp. adverb) to an adverb (resp. Adjective) by removing the string "able" (resp. "ingly") and adding the string "ingly" (resp. "able").
Definition 2: *a pseudo-suffix pair of a given language L is* **valid** *when the pseudo-suffixes involved are actual suffixes of the language L, and when the pair can be used to describe the passage from one*

word of a given derivational family of L to another word of the same family.

Two parameters are used to determine as precisely as possible valid pseudo-suffix pairs: the p-similarity and the number of occurrences of a pseudo-suffix pair. This last parameter accounts for the fact that the pseudo-suffix pairs encountered frequently are associated to actual suffixation processes whereas the less frequent ones either are associated to irregular phenomena or are not valid. But, in order to design a procedure which can be applied on several languages, and to avoid missing too many valid pseudo-suffix pairs, we have set these two parameters in the following loose way:

Definition 3: *a* **suffix pair** *of a language L is a pseudo-suffix pair of L which occurs more than once in the set of word couples of L which are at least 5-similar.*

Remarks:

- two words are at least k-similar if they are p-similar with $p \geq k$,
- all the suffix pairs are not valid. The above definition provides a set of pseudo-suffix pairs which is approximately contains the set of valid pseudo-suffix pairs. Our purpose here is not to miss any valid pseudo-suffix pair,
- the number of occurrences of a pseudo-suffix pair is set at 2, the minimal value one can think of, and which corresponds to our desire to remain language independent,
- the choice of the value 5 for the similarity factor represents a good trade off between the notion of long words and the desire to be language independent. We believe anyway that a slight change in this parameter won't lead to a set of pseudo-suffix pairs significantly different from the one we have.

Here is an example of French suffix pairs extracted from the French lexicon, with their number of occurrences:

ation+N	er+V	782
+AJ	ment+AV	460
eur+AJ	ion+N	380
er+V	on+N	50
sation+N	tarisme+N	5

All these suffix pairs are valid except the last one which is encountered in cases such as "autorisation - autoritarisme" (authorisation - authoritarianism). One can note that a valid suffix pair does not always link words which belong to the same derivational family. For example, the pair (er+V,on+N) yields the following link "saler - salon" (salt - lounge) though the two words refer to different concepts. The notion of validity only requires that two words of a same derivational family can be related by the suffix pair, which is the case for the previous pair in so far as it relates "friser - frison" (curl (+V) - curl (+N)).

3.2 Clustering words into relational families

The problem we have to face now is the one of grouping words which belong to the same derivational family, and to avoid grouping words which do not belong to the same derivational family. A simple idea one can try consists in adding words into a family to the extent they are p-similar, with a value of p to be determined, and related through suffix pairs. For example, given the two English suffix pairs (+V,able+AJ) and (+V,ment+N), we can first group the 6-similar words "deploy" and "deployable", and then add to this family the word "deployment". But such a procedure will also lead to group "depart" and "department" into the same family. The problem here is that suffix pairs relate words which do not belong to the same derivational family.

There is however one way we can try to automate the control of the removal of a suffix, based on the following intuitive idea. If the string "ment" is not a suffix, as in "department", then it is likely that the word obtained after removal of the string, that is "depart", will support suffixes which do not usually co-occur with "ment", such as "ure" which produces "departure". The underlying notion is that of suffix families, notion which accounts for the fact that the use of a suffix usually coincides with the use of other suffixes, and that suffixes from different families do not co-occur. Such an idea is used in (Debili, 1982), with manually created suffix families.

To take advantage of this idea, we used hierarchical agglomerative clustering methods. The following general algorithm can be given for hierarchical clustering methods:

1. identify the two most similar points (with similarity greater than 0)
2. combine them in a cluster
3. go back to step 1, treating clusters as points, till no more points can be merged (similarity 0)

Particular methods differ in the way similarity is computed. In our case, the initial points consist of words, and we define the similarity between two words, w_1 and w_2, as the number of occurrences of the suffix pair of L which links w_1 and w_2. If such a suffix pair does not exist, then the similarity equals 0. The similarity between clusters (or points as referred to in the above algorithm) depends on the method chosen. We tested 3 methods:

- single link; the similarity between two clusters is defined as the similarity between the two most similar words,
- group average; the similarity between two clusters is defined as the average similarity between words,
- complete link; the similarity between two clusters is defined as the similarity between the two less similar words,

The single link method makes no use at all of the notion of suffix families, and corresponds to the naive procedure described above. The group average method makes partial use of this notion, whereas the complete link heavily relies on it.

The clusters thus obtained represent an approximation of the derivational families of a language, and consitute our relational families.

Here is an exemple of some relational families obtained with the complete link method:

deprecate deprecation deprecator deprecative deprecativeness deprecatively deprecativity deprecatorily deprecatory deprecatingly

deposability deposable deposableness deposably depose deposer deposal

department departmentality departmental departmentalness departmentally

depart departure departer

3.3 Evaluation

We performed an evaluation on English considering as the gold reference a hand-built derivational lexicon that we have. We extracted derivational families from this lexicon, and compared them to the relational families obtained. This comparison is based on the number of words which have to be moved to go from one set of families to another set. Due to overstemming errors, which characterise the fact that some unrelated words are grouped in the same relational family, as well as to understemming errors, which correspond to the fact that some related words are not grouped in the same relational family, relational and derivational families often overlap.
To account for this fact, we made the assumption that a word w_i was correctly placed in a relational family r_i if this relational family comprised in majority words of the derivational family of w_i, and if the derivational family of w_i was composed in majority by words of r_i. That is there must be some strong agreement between the derivational and relational families to state that a word is not to move. All the words which did not follow the preceding constraints were qualified as “to move”. We directly used the ratio of words “not to move” to compute the proximity between relational and derivational families.

We, in fact, evaluated several versions of the relational families we built, in order validate or invalidate some of our hypotheses. The following table summarises the results obtained for the three clustering methods tested, with the parameters set as described above:

Single link	47%
Group average	77%
Complete link	85%

These results show the importance of the notion of suffix families, at least with the parameters we used. As a comparison, we performed the same evaluation with families obtained by two stemmers, the SMART and Porter’s stemmer, well-known in the Information Retrieval community. To construct families with these stemmers, we took the whole lemmatised lexicon, submitted it to the stemmers and grouped words which shared the same stem. We then ran the evaluation above and obtained the following results: SMART stemmer: 0.82 Porter’s stemmer: 0.65 Not surprisingly, the SMART stemmer, which is the result of twenty years of development, is a better approximation of derivational processes than Porter’s stemmer.

4 From relational to derivational morphology

Once the relational families have been constructed, they can be used to search for actual suffixes. Rather than performing this search directly from our lexicon, i.e. from all the possible word pairs, the clustering made to obtain word families allows us to restrict ourselves to a set of word pairs motivated by the broad notion of suffix we used in the previous section.

We thus use the following general algorithm, which allows us to estimate the parameters of the general probabilistic model given above:

- 1. from the lexicon, build relational families,
- 2. from relational families, build a set of word pairs and suffixes,
- 3. from this set, estimate some parameters of the general model,
- 4. use these parameters to induce a derivation tree on each relational family,
- 5. use these trees to refine the previous set of word pairs and suffixes, and go back to step 3 till an end criterion is found
- 6. the trees obtained can then be used to extract dependencies between suffixation operations, as well as morphographemic rules.

We will now describe steps 3, 4 and 5, and give an outline of step 6.

4.1 Extraction of suffixation operations

Since our lexicons contain neither phonological nor semantic information, the general probabilistic model given in the introduction can be simplified, so that it is based only on the graphemic and morpho-syntactic dimensions of words. Furthermore, since we restrict ourselves to concatenative languages, we adopt the following form for a suffixation operation S:

$$S = \begin{pmatrix} G_d = concat(G_o, s) \\ MS_o \rightarrow MS_d \end{pmatrix}$$

where G_d (MS_d) stands for the graphemic (morpho-syntactic) form of the derived word produced by the suffixation operation, G_o (MS_o) for the graphemic (morpho-syntactic) form of the original word on which the suffixation operation operates. *concat* is the concatenation operation, and s is the suffix associated to the suffixation operation S.

We can then write the probability that a word w_2 derives, through a suffixation process, from a word w_1 as follows:

$$P(w_1 \rightarrow w_2) =$$

$$= \sum_S p(S)p(G_1 \rightarrow G_2, MS_1 \rightarrow MS_2|S)$$

$$= \sum_S p(S)p(G_1 \rightarrow G_2|S)p(MS_1 \rightarrow MS_2|G_1 \rightarrow G_2, S)$$

$$\simeq \sum_S p(S)p(G_1 \rightarrow G_2|S)p(MS_1 \rightarrow MS_2|S)$$

the last equation being based on an independence assumption between the graphemic form and the morpho-syntactic information attached to words. Even though some morpho-syntactic information can be guessed from the graphical form of words, it is usually done via the suffixes involved in the words. Thus, conditioning our probabilities on the mere suffixation operations represents a good approximation to the kind of dependence that exists between graphemic form and morpho-syntactic information.

The term involving morpho-syntactic information, i.e. the probability to produce MS_2 from MS_1 knowing the suffixation operation S, can be directly rewritten as:

$$p(MS_1 \rightarrow MS_2|S) = \delta(MS_1, MS_o)\delta(MS_2, MS_d)$$

where δ is the Kronecker symbol ($\delta(x, y)$ equals 1 if the two arguments are equal and 0 otherwise).

The words we observe do not exactly reflect the different elements they are made of. Morphographemic rules, allomorphy and truncation phenomena make it difficult to identify the underlying structure of words (see (Anderson, 1992; Corbin, 1987; Bauer, 1983) for discussions on this topic). That is, the graphemic forms we observe are the results of different operations, concatenation being, in most cases, only the first.

Since: allomorphy, truncation and morphographemic phenomena do not depend on the words themselves but on some subparts of the words; direct concatenation gives a better access to the suffix used; and suffixation usually adds element to the original form[3], we use the following form for $p(G_1 \rightarrow G_2|S)$:

$$p(G_1 \rightarrow G_2|S) = 0 \; if \; l(G_1) \geq l(G_2)$$

$$else \; p(G_1 \rightarrow G_2|S) =$$

$$\begin{array}{ll} c_0 & if \; diff(G_1, G_2, s) = 0 \\ c_1 & if \; diff(G_1, G_2, s) = 1 \\ c_2 & if \; diff(G_1, G_2, s) = 2 \\ c_3 & if \; diff(G_1, G_2, s) = 3 \\ 0 & \quad otherwise \end{array}$$

where l(G) is the length of G, $diff(str1, str2, suff)$ represents the number of characters differing between $str1$ and $str2 - suff$ (i.e. the string obtained via removal of $suff$ from $str2$, proceeding backward from the end of $str2$), and $c_i, 0 \leq i \leq 3$ are arbitrary constants, the sum of which equals 1, which control the confidence we have on a suffix with respect to the edit distance between G_1 and G_2.

For our first experiments, we set the four constants c_0, c_1, c_2, c_3 according to the constraint:

$$c_3 = \frac{1}{2}c_2 = \frac{1}{4}c_1 = \frac{1}{8}c_0$$

which accounts for the fact that we give more weight to direct concatenation, then to concatenation with only 1 differing character, etc.

To estimate the probabilities $p(S)$, we first built a set of suffixation operations from relational families; for each word pair (w_1, w_2) found in a relational family, we consider all the suffixation operations S such that:

$$S = \begin{pmatrix} s \\ MS_1 \rightarrow MS_2 \end{pmatrix}$$

with s being a sequence of letters ending G_2 such that:

$$p(G_1 \rightarrow G_2|S) > 0$$

This process yields, besides the set of suffixation operations, a set of word pairs (w_1, w_2) possibly linked through a suffixation process. We will denote

[3]Due to truncation and subtraction, there may be cases where the derived form is shorter or the same length as the original form. However, these cases are not frequent, and should be recovered by the procedures which follow.

this last set by $\mathcal{WP}$. Some of the pairs in $\mathcal{WP}$ are valid, in the sense that the second element of the pair directly derives from the first element, whereas other pairs relate words which may or may not belong to the same family, through a set of derivational processes. However, since relational families represent a good approximation to actual derivational families, regular suffixation processes should emerge from $\mathcal{WP}$.

We then used the EM algorithm, (Dempster et al., 1977), to estimate the probabilities $p(S)$. Via the introduction of Lagrange multipliers, we obtain the following reestimation formula:

$$p_{\theta'}(S) =$$

$$\lambda^{-1} \sum_{\mathcal{WP}} \frac{p_\theta(S)p(G_1 \rightarrow G_2|S)p(MS_1 \rightarrow MS_2|S)}{\sum_{S'} p_\theta(S')p(G_1 \rightarrow G_2|S')p(MS_1 \rightarrow MS_2|S')}$$

where λ is a normalizing factor to assure that probabilities sum up to 1.

This method applied to French yields the following results (we display only the first 10 suffixes, i.e. the string s associated with the suffixation operation S, together with the POS of the original and derived words. The first number corresponds to the probability estimated):

0.071671	Noun →	**er**	→ Verb
0.019032	Adj →	**er**	→ Verb
0.018231	Verb →	**ion**	→ Noun
0.017365	Noun →	**ion**	→ Noun
0.017123	Noun →	**ur**	→ Noun
0.012864	Noun →	**eur**	→ Noun
0.011034	Noun →	**on**	→ Noun
0.010780	Noun →	**te**	→ Noun
0.009955	Adj →	**ation**	→ Noun
0.009881	Noun →	**nt**	→ Adj

As can be seen on these results, certain elements, such as **ur** are extracted even though the appropriate suffix is **eur**, our procedure privileging the element with direct concatenation (this concatenation happens after a word ending with an e). Note, however, that the true suffix is close enough to be retrieved.

4.2 Extraction of suffixal paradigms

The suffixes we extracted are derived from relational families. In these families, some words are related even though they do not derive from each other. The set of related words in a relational family defines a graph on this family, whereas the natural representation of a derivational family is a tree. We want to present here a method to discover such a tree.

A widely used tree construction method from a graph is the Minimal(Maximum) Spanning Tree method. We have adapted this method in the following way:

1. Step 1: for each word pair (w'_1, w'_2) in the family, compute $a = p(w'_1 \rightarrow w'_2)$,
2. Step 2: sort the pairs in decreasing order according to their a value,
3. Step 3: select the first pair (w_1, w_2), add a link with w_1 as father and w_2 as daughter,
4. Step 4: for each possible suffixation operation S such that $p(w_1 \rightarrow w_2|S) > 0$, add to the node w_1 the potential allomorph obtained by removing s from G_2, proceeding backward from the end of G_2,
5. Step 5: select the following pair, compute the set of allomorphs A, and add a link between the elements, if:
   (a) it does not create a loop,
   (b) if the first element of the pair, w'_1, is already present in the tree, then the set of allomorphs of w'_1 in the tree is either empty or has common elements with A. In the latter case, replace the set of allomorphs of w'_1 in the tree by its intersection with A,
6. Step 6: go back to Step 5 till all the pairs have been examined

This algorithm calls for the following remarks:

- we use allomorph in a broad sense, for lexemes: an allomorph of a word is simply a form associated to this word and which can be used as the support to derivation in place of the word itself,
- if two sets of allomorphs are not empty and do not have elements in common, then we face a conflict between which elements serve as a support for the different derivation processes. If they have common elements then the common elements can be used in the associated derivation processes. If one set is empty, then the word itself is used for one derivation process, and the allomorphs in the other.

Let us illustrate this algorithm on a simple exemple. Let us assume we have, in the same relational family, the three French words *produire (En. produce), production (En. production), producteur (En. producer).* Step 2 yields the two ordered pairs *(produire, production); (produire, producteur).* Steps 3 and 4 for the first pair provide the suffixes *(on, ion, tion, ction)* and the associated allomorphs for *produire*: *(produ, produc, product, producti).* When examining the pair *(produire, producteur)*, we obtain the suffixes *(ur, eur, teur, cteur)* with the allomorphs for *produire*: *(produ, produc, product).* The two sets of allomorphs have common elements. The final set of allomorphs for *produire* will obtained by intersecting the two previous sets, leading to: *(produ, produc,* *product)*. Note that the elimination of the form *producti* will lead to the rejection of the suffix *on* in subsequent treatment (namely the learning of suffixes from the trees, step 5 of the algorithm given at the beginning of section 2).

Once the trees have been constructed for all relational families (note that with our procedure, more than one tree may be used to cover the whole family), it is possible to reestimate the probabilities $p(S)$. This time, the word pairs are directly extracted from the trees, and, due to the sets of allomorphs, the probabilities $p(G_1 \rightarrow G_2|S)$ are not necessary anymore, since we will only rely on direct concatenation. Lastly, as described in the general algorithm, the new suffixation operations can be used again to build new trees, and so on and so forth, until an end condition is reached. A possible end condition can be the stabilization of the set of suffixation operations. Since our procedure gradually refines this set (at one iteration, the set of suffixation operations is a subset of the one used in the previous iteration), the algorithm will stop.

Another extension we can think of is the extraction, from the final set of trees, of morphographemic rules. Methods borrowed to Inductive Logic Programming seem good candidates for such an extraction, since these rules can be formulated as logical clauses, and since we can start from specific examples to the least general rule covering them (several researchers have addressed this problem, such as (Dzeroski and Erjavec, 1997)).

5 Conclusion

We have presented an unsupervised method to acquire derivational rules from an inflectional lexicon. In our opinion, the interesting points of our method lie in its ability to automatically acquire suffixes, as well as to induce a linguistically motivated structure in a lexicon. This structure, together with the elements extracted, can easily be revised and corrected by a lexicographer.

Acknowledgements

I thank two anonymous reviewers for useful comments on a first version of this paper.

References

G. Adamson and J. Boreham. 1974. The use of an association measure based on character structure to identify semantically related pairs of words and ocuments titles. *Information Storage and Retrieval*, 10.

S. R. Anderson. 1992. *A-morphous morphology.* Cambridge University Press.

L. Bauer. 1983. *English word-formation.* Cambridge University Press.

V. Cherkassky and F. Mulier. 1998. *Learning from data.* John Wiley and Sons.

D. Corbin. 1987. *Morphologie dérivationnelle et structuration du lexique.* Presses Universitaires de Lille.

W. Daelemans, J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 1999. Timbl: Tilburg memory based learner, version 2.0, reference guide. Technical report, ILK, Tilburg.

J. Dawson. 1974. Suffix removal and word conflation. *ALLC Bulletin.*

F. Debili. 1982. *Analyse syntaxico-sémantique fondée sur une acquisition automatique de relations lexicales-sémantiques.* Ph.D. thesis, Univ. Paris 11.

A. P. Dempster, N. M. Laird, and D. B. Dubin. 1977. Maximum likelihood from incomplete data via the em algorithm. *Royal Statistical Society*, 39.

S. Dzeroski and T. Erjavec. 1997. Induction of slovene nominal paradigms. In *Proceedings of 7th International Workshop on Inductive Logic Programming.*

B. Fradin. 1994. L'approche à deux niveaux en morphologie computationnelle et les développements récents de la morphologie. *Traitement automatique des langues*, 35(2).

M. Hafer and S. Weiss. 1974. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10.

C. Jacquemin and E. Tzoukerman. 1997. Guessing morphology from terms and corpora. In *Proceedings of ACM SIGIR.*

J.B. Lovins. 1968. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11.

C.D. Manning. 1998. The segmentation problem in morphology learning. In *Proceedings of New Methods in Language Processing and Computational Natural Language Learning.*

C. Paice. 1996. Method for evaluation of stemming algorithms based on error counting. *Journal of the American Society for Information Science*, 47(8).

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3).

A. Stolcke and S. Omohundro. 1994. Best-first model merging for hidden markov model induction. Technical report, ICSI, Berkeley.