Partially Supervised Learning of Morphology with Stochastic Transducers

Alexander Clark
ISSCO / TIM,
University of Geneva,
UNI-MAIL, Boulevard du Pont-d'Arve,
CH-1211 Genève 4,
Switzerland

Abstract

In this paper I present an algorithm for the unsupervised learning of morphology using stochastic finite state transducers, in particular Pair Hidden Markov Models. The task is viewed as an alignment problem between two sets of words. A supervised model of morphology acquisition is converted to an unsupervised model by treating the alignment as a further hidden variable. The use of the Expectation-Maximisation algorithm for this task is studied, which leads to calculations involving the *permanent* of a matrix of probabilities. The calculation of the permanent is discussed, and various technical difficulties are addressed. Results are presented on the English past tense and the Arabic plural, a highly complex system of non-concatenative morphology.

1 Introduction

A number of different approaches to the unsupervised learning of morphology have been presented in the past few years (Goldsmith, 2001; Schone and Jurafsky, 2000). Though they achieve impressive results, they share a common failing: an *a priori* limitation on the form of the morphological transductions that can be modelled, restricted to simple concatenation, and often only suffixation. This is clearly undesirable, since many non-Indo-European languages, and some Indo-European ones as well, such as German, use other inflectional processes, and it is precisely these less well studied languages that actually need automatic morphological inducers. A related limitation is that they can only learn regular morphology. Though these systems perform well within their limitations, a general language learning system cannot make these sorts of assumptions. Supervised learning algorithms on the other hand, are capable of learning irregularities and non-concatenative morphology (Rumelhart and McLelland, 1986; Mooney and Califf, 1995; Clark, 2001). What is desirable is to have a means for turning a supervised acquisition model into an unsupervised acquisition model.

This work is closely related to that of (Yarowsky and Wicentowski, 2000). They are concerned with integrating diverse sources of information; here I am concerned with the correct application of a single source of information, namely the surface forms of each word. As they say:

> But for many languages, and to a quite practical degree, inflectional morphological analysis and generation can be viewed primarily as an alignment task on a broad coverage word list.

The approach taken here is to use a stochastic model of the joint probability of the pair of strings – a supervised model, that can be trained with the EM algorithm – and to treat the alignment between the strings as a further hidden variable which can be modelled again with the EM algorithm. A clean and well-motivated treatment of this will allow integration of this into more complex and broader language acquisition systems.

Many other sources of information are available to aid in this alignment. The focus of this paper is to demonstrate a more rigorous method to apply one specific piece of information: the surface forms of the words to be aligned. In this paper I shall use a particular form of trainable stochastic finite-state transducer called the Pair Hidden Markov Model, but the approach could be used with other more powerful types of EM-trained transducers, such as context free transducers (Wu, 1997).

I shall not present a solution to the general problem of unsupervised learning of morphology here: I shall consider a slightly easier task, that I shall call *partially supervised* learning: this is where the learner is presented with two sets of strings, and must work out what the relationship is between them.

2 Pair Hidden Markov Models

Pair Hidden Markov Models (PHMMs) were introduced by Durbin *et al.* (1998) following on from much prior work in bioinformatics and used to learn morphology in (Clark, 2001). They are an extension of Hidden Markov Models (HMMs), that model the joint probability of a pair of strings, rather than a single string. Historically they derived as an extension of the Levenshtein edit distance, and are related to various kernels used in Support Vector Machines (Cristianini and Shawe-Taylor, 2000). They can be trained with modifications of the EM algorithm (Dempster et al., 1977), and are well suited to learning a range of transductions in natural language, including grapheme-phoneme conversion, but especially morphological transductions. A major advantage of them is that they can also learn non-concatenative transductions, such as vowel changes in the stem, a good example being the umlaut in German.

The difference between PHMMs and HMMs is that PHMMs can output symbols on two streams or sequences, which we can call the left and right streams. Thus instead of having the normal state conditional output functions, we have three sorts of outputs that can output the same symbol on both streams, a symbol on the left stream only or a symbol on the right stream only, which are called q_{11}, q_{10} and q_{01} transitions respectively. This is similar to the copy, delete and insert operations of the Levenshtein distance. This results in a model of the joint probability distribution of a pair of strings, which may be of different lengths. As discussed in (Ristad, 1997; Clark, 2001), the standard dynamic programming algorithms for HMMs can be modified to deal with PHMMs. In particular an EM based parameter estimation algorithm similar to the forward-backward algorithm can be applied to learn the transductions. A related but more general algorithm has been presented in (Eisner, 2001).

Because of the fact that the q_{11} transitions output the same symbol on both streams, randomly initialised models have a strong bias to produce similar strings on both left and right streams. We can use this bias to align similar strings.

3 Perfect Situation

I will start with an artificially simple situation. Let us suppose we have two sets of strings U and V of the same size. We wish to align them, i.e. find a bijection between them, and simultaneously train a model on the aligned data. We can model this as a stochastic process in two stages: first we generate n pairs of strings, and then we generate a permutation of the second set that shuffles them. We can model the permutation as a hidden variable X, that takes one of the $n!$ permutations as its value. We can consider this as an $n \times n$ permutation matrix such that $X_{ij} = 1$ if the ith element of U is aligned with the jth element of V and is zero otherwise.

$$p(U, V) = \sum_X p(X)p(U, V|X) \qquad (1)$$

Since we have no reasons to prefer one permutation rather than another we set $p(X) = \frac{1}{n!}$. The probability given the alignment is just the product of the probabilities of the matching pairs,

$$p(U,V|X) = \prod_i p(u_i, v_{X(j)}) \qquad (2)$$

If we consider the matrix P which has in i,j the element $p(u_i, v_j)$, the sum of the $n!$ permutations is called in linear algebra the *permanent* of the matrix (Bhatia, 1996).

$$Per(P) \stackrel{def}{=} \sum_\sigma \prod_i P_{i,\sigma(i)} \qquad (3)$$

where σ ranges over all the permutations of n elements.

It is similar to the more familiar determinant but without the alternating signs. One problem is that it is not possible to calculate this efficiently (Barvinok, 1999). There is a great deal of active research in this area though as it is possible to encode various combinatorial problems into an appropriate matrix. [1]

To train the model with the EM algorithm we need to be able to calculate the posterior expectation of X_{ij} given the data and the model. Since X_{ij} is one or zero, the expectation equals the probability that i is aligned with j.

$$p(X_{ij}|U,V) = \frac{\sum_{X:X_{ij}=1} p(U,V|X)}{\sum_X p(U,V|X)} \qquad (4)$$

The denominator of this fraction is the permanent of the matrix P, i.e. the sum over all $n!$ permutations. The numerator is the sum of the $(n-1)!$ of those permutations that have $X_{ij} = 1$. This is the product of P_{ij} with the permanent of the ij-minor [2] of P.

Thus we can write

$$E[X_{ij}|U,V] = \frac{P_{ij}Perm(P^{ij})}{Perm(P)} \qquad (5)$$

We can consider these posterior probabilities as a matrix, that will be doubly stochastic. This map from the matrix of probabilities to the matrix of posteriors is sometimes called the Bregman map (Bregman, 1967). Though intractable to compute exactly, we can approximate it under certain circumstances using the technique of Sinkhorn balancing (Sinkhorn, 1964), as advocated by (Beichl and Sullivan, 1999). This converges rapidly (Soules, 1991) giving a overall complexity of $\mathcal{O}(n^3)$, which is tractable for matrices with dimensions ≈ 1000, such as we use here. The method of Sinkhorn balancing is intuitively quite straightforward; we want to scale a positive matrix so it is doubly stochastic. If we normalise the row sums, we will have a matrix that is row-stochastic, i.e. has its row sums equal to unity, but not necessarily column-stochastic. If we then normalise the column sums, we will have a matrix that is column-stochastic but probably not row-stochastic. If we continue in this way, alternating normalising the rows and normalising the columns, we converge to a doubly stochastic matrix which under certain circumstances is an approximation to the Bregman map. We now have the basis for an algorithm.

1 Choose a random model.

2 Calculate the matrix of $p(u_i, v_j)$

3 Estimate the matrix of the posteriors, using Sinkhorn balancing.

4 Train the model on every pair (u_i, v_j) weighting by the value of the posterior probability.

5 Repeat from step 2, until the posterior probability matrix is (very close to) a permutation matrix.

Theoretically we know this will converge by the EM algorithm and initial experiments with this framework showed that the matrix of posteriors rapidly converges to a permutation matrix.

4 Imperfection

This is obviously a highly artificial situation. More interesting cases are where we have two sets that are not the same size, with two subsets that we want to align. More formally we have two sets of strings U and V of size m

[1] For example, finding the number of maximum matchings in a bipartite graph.

[2] The ij-minor of a matrix is the matrix formed by removing the row and column containing the ij element, to form an $n-1$ by $n-1$ matrix.

and n respectively with subsets $U' \subset U$ and $V' \subset V$ both of size k. We then have three models. A model for U and a model for V and a model for a joint distribution over $U' \times V'$. We then have a hidden variable which corresponds to the selection of the sets and the bijection between U' and V'. Given a value of X we can write the total likelihood function as:

$$p(U,V|X) = \prod_{u \in U-U'} p_U(u) \prod_{v \in V-V'} p_V(v) \left(\prod_{u \in U', v \in V'} p_M(u,v) \right)^{\alpha} \tag{6}$$

We now have a certain degree of flexibility in how we define $p(X)$; we can make it depend on the size of the sets that are selected. We can use this to make some of the calculations more tractable.

This algorithm allows one to trade off the gain from aligning them against not aligning them, by including models for the information in the individual sequences (Allison et al., 1999). We can tweak it if need be by raising the joint model probability to a power $\alpha \in [1,2]$. This will have the effect of making it less likely to align words; if α is 1, then it is likely to align words even when they have little relation since $p_M(u,v)$ can in general be at least $p_U(u)p_V(v)$ for related u and v. Conversely, values of α close to 2, will mean that the model will only align the words if there is a very strong link between them. Thus α is a tunable parameter that allows us to adjust the recall/precision trade-off.

Suppose U has m elements, and V has n elements, then we have an $m \times n$ matrix of the joint probabilities, which we can call P_M We can also define a $m \times m$ diagonal matrix, corresponding to the probabilities according to the model of U, which we can call P_U, and a $n \times n$ diagonal matrix for the probabilities of V, P_V. If we also create a matrix of size $n \times m$ with every element 1, P_1 then we can form an $(m+n)$ square matrix thus:

$$M = \begin{pmatrix} P_U & P_1 \\ P_M & P_V \end{pmatrix} \tag{7}$$

Then every permutation of this matrix corresponds to a particular choice of the alignment, and we can use exactly the same techniques for estimating the posterior probabilities on this matrix, as we did before. There is one substantive difference which is that because of the block of ones in the top right, alignments that align k of U and V together will have a "bonus" factor of $k!$, corresponding to the $k!$ paths through the $k \times k$ submatrix of P_1. This is generally good, since we want to encourage the algorithm to align as much as possible. We can accomodate this formally by making $p(X)$ in our generative model be proportional to $k!$. We then train all three models, weighting the probabailities by the appropriate values from the posterior matrix.

I will take a simple example: suppose $U = \{\mathrm{cat}, \mathrm{dog}, \mathrm{fox}\}$ and $V = \{\mathrm{cats}, \mathrm{dogs}\}$. Table 1 shows the resulting composite matrix.

Now each permutation of this matrix will correspond to a particular alignment, and the probability given that alignment will be the product of the appropriate elements of the matrix. The identity matrix will correspond to none being aligned, and will have probability equal to the product of the elements along the diagonal of the matrix in Table 1, i.e.

$$p(U,V|X) = p_U(\mathrm{cat}) p_U(\mathrm{dog}) p_U(\mathrm{fox}) \, p_V(\mathrm{cats}) p_V(\mathrm{dogs}) \tag{8}$$

If *cat* and *dog* are aligned correctly then (k=2) there are 2!, matrices which are shown here

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{9}$$

and also

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{10}$$

each of which will have probability

$$
\begin{pmatrix}
p_U(\text{cat}) & 0 & 0 & 1 & 1 \\
0 & p_U(\text{dog}) & 0 & 1 & 1 \\
0 & 0 & p_U(\text{fox}) & 1 & 1 \\
p_M(\text{cat,cats}) & p_M(\text{dog,cats}) & p_M(\text{fox,cats}) & p_V(\text{cats}) & 0 \\
p_M(\text{cat,dogs}) & p_M(\text{dog,dogs}) & p_M(\text{fox,dogs}) & 0 & p_V(\text{dogs})
\end{pmatrix}
$$

Table 1: Matrix of probabilities for $U = \{\text{cat}, \text{dog}, \text{fox}\}$ and $V = \{\text{cats}, \text{dogs}\}$.

$$p(U, V|X) = p_M(\text{cat,cats})p_M(\text{dog,dogs})p_U(\text{fox}) \tag{11}$$

5 Experiments with English

In all of the experiments that follow I used a PHMM to model the joint probability and two HMMs to model each of the two left-over sets. All three have the same number of states. I ran the models for 5-10 iterations after which the models has effectively converged, and considered two words to be aligned if the posterior probability was greater than 0.1, though in fact the values of aligned pairs were invariably very close to 1.

To experiment with this approach I chose two contrasting morphological processes. First, I chose the English past tense, which has been studied extensively. This has several advantages: first, it is very well known and understood, and the correct analyses are not in dispute; secondly, though it is rather simple, it is by no means trivial; and thirdly, there are standard test sets available which facilitate comparison with other techniques.

The data set I used here (Ling, 1994) consisted of a set of 2163 base forms with 1394 past forms, in the UNIBET phonetic alphabet. Table 2 summarizes the results of the algorithm on the various data sets. When trained with a very small 5 state transducer, the algorithm produced a highly accurate alignment. Analysis of the resulting transducer showed that it was not correctly selecting the suffix to be applied, but nonetheless was modelling the process in a crude way. Table 3 shows the behaviour of the model when applied to the (unseen) nonce word *gagaga*. The model is not sensitive to the phonology of the stem.

Past form	$p(v\|u)$	$p(u\|v)$
gagagad	0.50	0.82
gagagat	0.15	0.78
gagagaId	0.023	0.72

Table 3: The three regular suffixes in the English past tense, with both conditional probabilities for the nonce word *gagaga*. Note that though it cannot select the correct suffix given the base form, given the inflected form, it can recover the base form. This is with a 5-state model.

The second experiment on English was with a noisier data set derived using an unsupervised technique. A slightly simplified version of the Wall Street Journal corpus was used, which had been 'speechified' as in (Charniak, 2001). An unsupervised clustering technique described in (Ney et al., 1994) was used to cluster the words into 64 classes. Two classes that corresponded to singular and plural nouns were manually selected producing a test set with 680 words in the singular noun class, and 1151 words in the plural class. In this data set there were 314 matching pairs of nouns in singular and plural form, together with substantial amounts of noise. The performance of the algorithm on this data set, denoted by Ney, is summarised in Table 2.

6 Experiments with Arabic

Arabic is a language with a complex system of non-concatenative morphology, with the Arabic broken plural being of particular complexity and interest (McCarthy and Prince, 1990). Space does not permit more than a brief summary: Arabic is based in general around lexical roots with three or four consonants, with the pattern of vowels providing morphosyn-

Data	States	α	U	V	Pairs	Correct	Incorrect	Precision	Recall
Ling	5	1.0	2163	1394	1394	1324	33	97.6	95.3
McCarthy	10	1.0	2450	3101	2450	1654	795	67.5	67.5
Plunkett1	10	1.0	859	859	859	820	27	96.8	95.5
Plunkett2	10	1.0	443	430	215	157	405	38.8	73.8
Plunkett2	10	1.5	443	430	215	140	25	84.8	65.1
Plunkett2	10	1.75	443	430	215	78	2	97.5	36.2
Plunkett2	10	2.0	443	430	215	2	0	100.0	0.9
Ney	5	1.0	680	1151	316	310	369	45.7	98.1
Ney	5	1.5	680	1151	316	306	52	85.5	96.8
Ney	5	1.75	680	1151	316	284	7	97.6	89.9
Ney	5	2.0	680	1151	316	276	2	99.2	87.3
Ney	10	1.75	680	1151	316	286	12	96.0	90.5

Table 2: Comparison of results on the various test sets. The second set of results on the Plunkett2 data set show the effect the exponent has on the precision and recall.

tactic information. The singular is changed to the plural by changes to the vowels and insertions of other consonants, that are sensitive to the overall prosodic outline of the root. There is also a *sound* plural that is formed by a more conventional process of suffixation. The first data set (MCCARTHY) was used in (McCarthy and Prince, 1990), and consists of 2450 singular nouns and 3101 plurals, in a fully vocalised phonemic transcription taken from the Wehr dictionary of Modern Standard Arabic. In this case the difference in numbers arises because many of the singular forms have multiple possible plurals. The data set is quite noisy as well, and in addition has a number of cases like this:

Singular	Plural
`?azabb`	`zubb`
`zubb`	`?azbaab`

Clearly, algorithms of the sort I use here will tend to align `zubb` with `zubb`. Moreover there are other complex morphological processes at work (one can form the plural of a plural). I trained a 10-state model on this data set (McCarthy). The overall results were quite poor. Error analysis revealed that the alignment had correctly identified that the consonant root system was the defining characteristics, but this was not enough to allow the alignment to proceed accurately.

I then experimented with a slightly simplified data set prepared for (Plunkett and Nakisa, 1997); this is substantially smaller – consisting of 859 pairs, drawn from the same dictionary but consisting of a mixture of sound and broken plurals, with only one plural form for each singular form. I performed two experiments: one with the data set as it was (Plunkett1), and one where I randomly removed half of the singulars and half of the plurals, to see whether the algorithm could correctly match up in the presence of the missing data. This produced a data set (Plunkett2) of size 443/430, with 215 possible pairs to be aligned. The results of these three tests are summarised in Table 2. This second data set, I ran with various values of the exponent α to see what effect it has on precision and recall. As expected, the algorithm performed well on the initial data set, aligning with high accuracy and precision. On the imperfect data set, Plunkett2, the effect of the exponent is quite marked. With the exponent at 1.0, the precision is very poor, but as the exponent is increased the precision increases rapidly. Note that it would be possible to repeatedly apply the algorithm to the same data, removing at each time the pairs already aligned, thus combining a number of high precision models into a high recall one. The small number of states employed are for efficiency purposes, they are clearly too small to allow correct modelling of these processes.

There is very little work that is directly comparable: there has been no prior work on the unsupervised learning of Arabic. However we can compare the results in English to (Yarowsky and Wicentowski, 2000). They use a number of different sources of information, and have results ranging from 31.3% using only Levenshtein distance after 1 iteration to 99.2 % for the final model combining all sources of information including frequency and semantic information.

7 Discussion

The motivation for this work is two-fold: first, it is clearly desirable for engineering reasons to be able to extract the morphology of a language automatically from a language rather than having to manually construct it. Secondly, a key area of cognitive science is the modelling of language acquisition. In many languages it may be possible to learn the alignments between words based on semantic information, in languages which are highly inflected the individual token frequencies may be too low to allow this to work. In these cases some sort of aligment operation must be performed. Completely unsupervised algorithms seem unnecessary: it is possible to extract sets of syntactically similar words from corpora using unsupervised clustering algorithms (Brown et al., 1992), and then to apply algorithms of this type to the result.

Better understanding of the correct application of these techniques is necessary to allow the correct treatment of morphologically rich languages such as Hungarian or Finnish.

(De Roeck and Al-Fares, 2000) presents an algorithm for identifying Arabic roots, that uses a language specific distance function; similarly (Yarowsky and Wicentowski, 2000) use a weighted edit distance as one component of their model, also performing a Viterbi approximation to the EM reestimation; for particular languages it will always be able to perform well with a simpler algorithm using prior knowledge about the language in question. This is clearly not an option in cognitive modelling, since it must work with all languages without language-specific information. [3] These techniques could also be used as a fairly general algorithm for satisfying combinatorial constraints, related to other work in QAP-optimisation (Gold and Rangarajan, 1996).

The algorithms presented here are comparatively slow in their naive form since we have to compute all the elements of the matrix, so it is $\mathcal{O}(|U||V|)$. A simple optimisation could be used to avoid having to compute pairs that are obviously not related. Of course, *went* is *obviously* not the past tense of *go*, so this approach will introduce errors. The next step is to move this to a completely unsupervised algorithm, and to integrate it with other components of a complete language acquisition system. In particular, in order to handle complete stem suppletion it seems likely that other sorts of information will be required: frequency and semantic information are the obvious candidates. Since suppletion tends to occur infrequently and with very frequent words, these should suffice. This may allow an understanding of the prevalence of phonological transparency in natural languages.

Acknowledgements

I am grateful to Alexander Barvinok for pointing out to me the work of Beichl et al. I am also grateful to John McCarthy and Ramin Nakisa for allowing me to use their data. Thanks also to Bill Keller, Eric Gaussier and others for helpful comments. Part of this work was done as part of the TMR network *Learning Computational Grammars*.

References

L. Allison, D. Powell, and T. I. Dix. 1999. Compression and approximate matching. *The Computer Journal*, 42(1):1–10.

A. I. Barvinok. 1999. Polynomial time algorithms to approximate permanents and mixed discriminants within a simple exponential factor. *Random Structures and Algorithms*, 14:29–61.

Isabel Beichl and Francis Sullivan. 1999. Approximating the permanent via importance sampling

[3] Though nativists might wish to provide information about Universal Grammar.

with application to the dimer covering problem. *Journal of Computational Physics*, 149(1):128–147.

Rajendra Bhatia. 1996. *Matrix analysis*. Springer.

L. M. Bregman. 1967. Proof of convergence of Sheleikhovskii's method for a problem with transportation constraints. *Zh. vychsl. Mat. mat. Fiz.*, 147(7).

Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souza, Jenifer C. Lai, and Robert Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Eugene Charniak. 2001. Immediate head parsing for language models. In *Proceedings of the 39th annual meeting of the ACL*, pages 116–123, Toulouse, France.

Alexander Clark. 2001. Learning morphology with Pair Hidden Markov Models. In *Proceedings of the Student Workshop at the 39th Annual Meeting of the Association for Computational Linguistics*, pages 55–60, Toulouse, France, July.

Nello Cristianini and John Shawe-Taylor. 2000. *Support Vector Machines*. Cambridge University Press.

Anne N. De Roeck and Waleed Al-Fares. 2000. A morphologically sensitive clustering algorithm for identifying Arabic roots. In *COLING-2000*.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, 39:1–38.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. 1998. *Biological Sequence Analysis: Probabilistic Models of proteins and nucleic acids*. Cambridge University Press.

Jason Eisner. 2001. Expectation semi-rings: Flexible EM for learning finite-state transducers. In *Proceedings of the ESSLLI Workshop on Finite-State Methods in NLP*, Helsinki, August.

Steven Gold and Anand Rangarajan. 1996. Softmax to softassign: Neural network algorithms for combinatorial optimization. *Journal of Artificial Neural Networks*, 2(4):381–399, Aug.

John A. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

Charles X. Ling. 1994. Learning the past tense of English verbs: The symbolic pattern associator vs. connectionist models. *Journal of Artifical Intelligence Research*, 1:209–229.

J. McCarthy and A. Prince. 1990. Foot and word in prosodic morphology: The Arabic broken plural. *Natural Language and Linguistic Theory*, 8:209–284.

Raymond J. Mooney and Mary Elaine Califf. 1995. Induction of first-order decision lists: Results on learning the past tense of English verbs. *Journal of Artificial Intelligence Research*, 3:1–24.

Hermann Ney, Ute Essen, and Reinhard Kneser. 1994. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–38.

Kim Plunkett and Ramin Charles Nakisa. 1997. A connectionist model of the Arabic plural system. *Language and Cognitive Processes*, 12(5/6):807–836.

Eric Sven Ristad. 1997. Finite growth models. Technical Report CS-TR-533-96, Department of Computer Science, Princeton University. revised in 1997.

D. E. Rumelhart and J. L. McLelland. 1986. On learning past tenses of English verbs. In D. E. Rumelhart and J. L McLelland, editors, *Parallel Distributed Processing*, volume 2, pages 216–271. MIT Press, Cambridge, MA.

Patrick Schone and Daniel Jurafsky. 2000. Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 67–72, Lisbon, Portugal.

R. Sinkhorn. 1964. A relation between arbitrary positive matrices and doubly stochastic matrices. *Annals of Mathematical Statistics*, 35(2).

G. W. Soules. 1991. The rate of convergence of Sinkhorn balancing. *Linear Algebra and Its Applications*, 150(3).

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, September.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of ACL 2000*, pages 207–216, Hong Kong.