Learning Morphology with Pair Hidden Markov Models

Alexander Clark
Cognitive and Computing Sciences,
University of Sussex,
Brighton BN1 9QH,
United Kingdom
alexc@cogs.susx.ac.uk

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

Abstract

In this paper I present a novel Machine Learning approach to the acquisition of stochastic string transductions based on Pair Hidden Markov Models (PHMMs), a model used in computational biology. I show how these models can be used to learn morphological processes in a variety of languages, including English, German and Arabic. Previous techniques for learning morphology have been restricted to languages with essentially concatenative morphology.

1 Introduction

As has been known for some time (Kaplan and Kay, 1994), finite-state methods are in large part adequate to model morphological processes in many languages. In this paper I present algorithms for learning sets of finite-state transducers from pairs of uninflected and inflected words. This approach differs from two-level morphology (Koskenniemi, 1983) in that it is concerned with transductions between surface strings rather than between a deep lexical string and a surface string. The techniques presented here are, however, applicable to learning other types of string transductions.

This paper is structured as follows: in the first part, sections 2 and 3, I introduce PHMMs and briefly discuss the modifications necessary to the standard HMM algorithms. In the second part, sections 4, 5 and 6, I present a model for morphology acquisition using PHMMs, and in the third part I present three experiments on learning morphology in English, German and Arabic. I conclude with a brief discussion.

2 Pair Hidden Markov Models

The algorithm used in this paper is based on Pair Hidden Markov Models (PHMMs). These were were introduced by Durbin *et al.*(1998) as a way of representing various alignment algorithms in bioinformatics. They are closely related to Hidden Markov Models (HMMs) as used in many NLP applications. Instead of outputting symbols in a single stream, however, they output them on two separate streams, the *left* and *right* streams. At each transition the PHMM may output the same symbol on both streams, a symbol on the left stream only, or a symbol on the right stream only. I call these q_{11}, q_{10} and q_{01} outputs respectively. For each state s the sum of all these output parameters over the alphabet A must be one.

$$\sum_{c \in A} q_{11}(c|s) + q_{10}(c|s) + q_{01}(c|s) = 1$$

Since we are concerned with finite strings rather than indefinite streams of symbols, we have in addition to the normal initial state s_0, an explicit end state s_1, such that the PHMM terminates when it enters this state. The PHMM then defines a joint probability distribution on pairs of strings from the alphabet. Though we are more interested in stochastic transductions, which are best represented by the conditional probability of one string given the other, it is more convenient to operate with models of the joint probability,

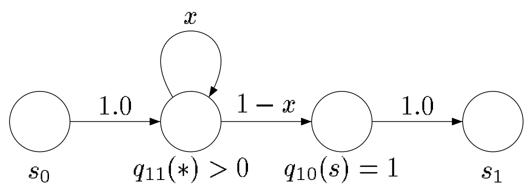

Figure 1: Example of a PHMM that adds an s to the end of any non-empty string.

and then to derive the conditional probability as needed later on.

Figure 1 shows a simple PHMM that adds an s onto the end of any string: more formally the joint probability of a pair of strings is greater than zero if and only if the second string is the concatenation of the first string with the string s. We can consider the model as operating either from the left stream to the right stream (adding an s) or from the right to the left (removing an s).

Note that since the q_{11} output emits the same symbol on each stream, a transduction which changes a symbol (say a to b) must use two transitions, first outputting a on the left alone and then a b on the right, or alternatively in the other order.

3 Parameter estimation

It is possible to modify the normal dynamic-programming training algorithm for HMMs, the Baum-Welch algorithm (Baum and Petrie, 1966) to work with PHMMs as well. This algorithm will maximise the joint probability of the training data.

We define the *forward* and *backward* probabilities as follows. Given two strings $u_1, \dots u_l$ and $v_1, \dots v_m$ we define the forward probabilities $\alpha_s(i,j)$ as the probability that it will start from s_0 and output $u_1, \dots, u_i$ on the left stream, and $v_1, \dots, v_j$ on the right stream and be in state s, and the backward probabilities $\beta_s(i,j)$ as the probability that starting from state s it will output $u_{i+1}, \dots, u_l$, on the right and $v_{j+1}, \dots, v_m$ on the left and then terminate, ie end in state s_1.

We can calculate these using the following recurrence relations:

$$\alpha_s(i,j) = \sum_{s'} \alpha_{s'}(i,j-1)p(s|s')q_{01}(v_j|s) + \sum_{s'} \alpha_{s'}(i-1,j)p(s|s')q_{10}(u_i|s) + \sum_{s'} \alpha_{s'}(i-1,j-1)p(s|s')q_{11}(u_i,v_j|s)$$

$$\beta_s(i,j) = \sum_{s'} \beta_{s'}(i,j+1)p(s'|s)q_{01}(v_{j+1}|s') + \sum_{s'} \beta_{s'}(i+1,j)p(s'|s)q_{10}(u_{i+1}|s') + \sum_{s'} \beta_{s'}(i+1,j+1)p(s'|s)q_{11}(u_{i+1},v_{j+1}|s')$$

where, in these models, $q_{11}(u_i,v_j)$ is zero unless u_i is equal to v_j. Instead of the normal two-dimensional trellis discussed in standard works on HMMs, which has one dimension corresponding to the current state and one corresponding to the position, we have a three-dimensional trellis, with a dimension for the position in each string. With these modifications, we can use all of the standard HMM algorithms. In particular, we can use this as the basis of a parameter estimation algorithm using the expectation-maximisation theorem. We use the forward and backward probabilities to calculate the expected number of times each transition will be taken; at each iteration we set the new values of the parameters to be the appropriately normalised sums of these expectations.

Given a PHMM, and a string u, we often need to find the string v that maximises $p(u,v)$. This is equivalent to the task of finding the most likely string generated by a HMM, which is NP-hard (Casacuberta and de la Higuera, 2000), but it is possible to sample from the conditional distribution $p(v|u)$, which allows an efficient stochastic computation. If we consider only what is output on the left stream, the PHMM is equivalent to a HMM with null transitions corresponding to the q_{01} transitions of the PHMM. We can remove these using standard techniques and then use this to calculate the *left backward* probabilities for a particular string u: $\beta^L_s(i)$

defined as the probability that starting from state s the PHMM generates $u_{i+1}, \ldots, u_l$ on the left and terminates. Then if one samples from the PHMM, but weights each transition by the appropriate left backward probability, it will be equivalent to sampling from the conditional distribution of $P(v|u)$. We can then find the string v that is most likely given u, by generating randomly from $p(v|u)$. After we have generated a number of strings, we can sum $p(v|u)$ for all the observed strings; if the difference between this sum and 1 is less than the maximum value of $p(v|u)$ we know we have found the most likely v. In practice, the distributions we are interested in often have a v with $p(v|u) > 0.5$; in this case we immediately know that we have found the maximum.

4 Paradigms in Morphology

Natural language morphology cannot and should not be represented as a simple string transduction from uninflected to inflected form. Which morphological processes a word undergoes are in general specified lexically not phonologically; different words with stems that are homophonous may take different inflections. Even English, with its rather impoverished morphology provides various illustrations of this – there are three verbs in English that are homophonous with *ring*: *ring* which takes the past tense *rang*, *wring* which takes the past tense *wrung* and the denominal *ring*, meaning *put a ring around* which forms the regular past tense *ringed*. That is, the choice of inflection must depend not just on the stem, but also on the particular lexical item concerned. Moreover, for computational reasons it is not possible to compute a single enormous PHMM with a thousand states that completely represents a particular morphological process.

It is therefore natural to decompose the problem into a mixture of separate PHMMs, where each PHMM will represent the transduction associated with a particular morphological class. A mixture of PHMMs

$$\sum_{i=1}^{k} p(i)p_i(u,v)$$

where $p(i)$ is the prior probability of the class i, and $p_i(u,v)$ is the joint probability of the pair u, v with respect to the ith model, is mathematically equivalent to a larger PHMM with a block diagonal transition matrix.

We consider the model then to be made up of k separate PHMMs, and all of the stems to be divided into k morphological classes or paradigms that correspond to the declensions or conjugations of the words. The transduction then applies the appropriate PHMM to each lexical item. We represent this in the model by having parameters for each paradigm and word $p(u|i)$ representing the probability of the word given the paradigm.

The overall joint distribution defined by the model will then be

$$\sum_{i=1}^{k} p(i)p(u|i)p_i(v|u)$$

5 Smoothing

A problem with the approach described above is that it would leave no probability mass for words not in the training data. We therefore smooth this model, using a simple weight for each class, changing the formula to:

$$\sum_{i=1}^{k} p(i)\left(\lambda_i p(u|i)p_i(v|u) + (1-\lambda_i)p_i(u,v)\right)$$

We can estimate the values of the λ_i smoothing parameters by using the EM algorithm on held-out data in the normal way. Intuitively, what these parameters tell us is how productive each paradigm is. If the λ_i is 1 this will mean that the paradigm is not productive, whereas values less than one will mean that unseen words using this paradigm were found in the held out data, and thus the paradigm is productive.

For a previously unseen word u_w, the probability $p(u_w|i)$ will be zero. Therefore the paradigm used will be the productive paradigm which is most likely to generate u_w on the left. Because the models define joint probabilities, when we sum the distributions, the one that applies will be the one that gives the highest probability to the stem.

In addition to smoothing the parameters that combine the models, it is also necessary to smooth the PHMMs themselves. This is done using standard interpolation techniques. We smooth only the output functions. We define three output functions, $q^*_{01}, q^*_{10}, q^*_{11}$ by tying all transitions of the three types : and for each state i in each model we define four parameters , and define the new output function $\hat{q}_i$ for each state to be:

$$\hat{q}_i = \lambda^i q_i + \lambda^i_{01} q^*_{01} + \lambda^i_{10} q^*_{10} + \lambda^i_{11} q^*_{11}$$

where for each state $\lambda^i + \lambda^i_{01} + \lambda^i_{10} + \lambda^i_{11} = 1$.

6 Description of the Algorithm

The algorithm proceeds incrementally. Rather than trying to model all of the data at once, we add PHMMs one at a time. At each iteration, we have a set of lexical items that we have not modelled. During the training of the model, we weight each pair by the conditional probability with respect to the current model. This allows the PHMM to select which pairs it will model. Once the PHMM has converged on a subset of the data, we then train a larger model (with five more states) on this subset. This is necessary because the minimal model that performs a particular transduction may be too small to correctly capture the phonological context that determines whether that transduction applies or not. If no data is modelled correctly , we increase the number of states by one. The algorithm starts with the minimal number of states which is three: an initial state, a final state and a single internal state. The only transduction that a model with this number of states can model is the identity transduction. This process is repeated, until all of the training data has been modelled. This produces a series of models, starting with large regular classes containing many data points, and ending with irregular classes modelling a single pair. Finally, calculate the smoothing parameters on held out data.

7 Experiments on the English past tense

The English past tense in spite of its simplicity has become something of a test case for the modelling of the acquisition of morphology. I used a standard data set,(Ling, 1994) which consisted of 1834 pairs of uninflected and inflected verb forms in the UNIBET phonemic transcription [1]. I used the associated Kucera-Francis frequency information to define a simple distribution, and sampled 20,000 tokens each for the training, held-out and test data. This evaluation corresponds to the situation the language learner is in – the training data is likely to have nearly all the irregular verbs in, and thus the generalisation ability of the algorithm is measured mostly on regulars. When testing, I considered a pair to be correct if the conditional probability of the inflected form given the uninflected form was greater than 0.5 – in this case it is clear that it is also the most likely inflected form to be produced by the transduction.

The algorithm produced 28 classes. Table 1 shows some of the classes obtained. Note that the three regular suffixes are assigned to different classes. The remaining classes correspond in general to the traditional classes of irregular verbs, though in some cases more than one class is modelled by the same PHMM, and some members of classes end up in a different class. The first cluster consists of all and only the verbs where the past is the same as the base form.

Results on test data were out of 20000 tokens, 19991 correct which is (99.96%), and out of 1571 types, 1567 correct (99.74%). 123 of these types were not in the training data. The four errors were *withhold*, *thrust*, *bind*, and *ring*. The first three were over-regularised and the last one was given the irregular plural "rang". The correct answer was stored in the model, but not accessed because of the homophony of the stem. These results are probably as good as can be expected with this methodology, and only possible because

[1] I use normal orthography for the examples in this paper to improve readability.

Table 1: Classes derived from the English past tense. Some have been removed for reasons of space. Note that the only productive paradigms are the three regular suffixes.

i	States	Words		λ
0	6	22	bet, shed	1.0
1	8	727	$+d$	0.90
2	8	281	$+t$	0.89
3	10	434	$+ed$	0.92
4	10	1	fly	1.0
5	12	26	break, draw	1.0
...				
24	20	2	sell, tell	1.0
25	20	2	go, undergo	1.0
26	22	1	leave	1.0
27	22	1	lose	1.0

of the extreme regularity and predictability of the English past tense. The model shows excellent generalisation, together with correct memorisation of the irregular exceptions.

8 German and Arabic plurals

I will discuss briefly some experiments on the learning of the plural in German and Arabic. German has a much more complex system of noun inflection(Cahill and Gazdar, 1999). The data for this experiment was taken from the CELEX lexical database. I extracted all pairs of singular and plural nouns in the CELEX phonetic alphabet, together with frequency information, creating a data set with 17076 pairs. I sampled 20,000 tokens to create a training set with 4709 pairs. Given that the declension of German nouns is not predictable from the phonology, it is inappropriate to evaluate it on the basis of its performance on unseen data.

The final model divided the data into 73 classes. Table 2 shows the seven largest classes created by the algorithm. Note that the model for class 5 is a nonconcatenative transduction mapping including *kraft*$\rightarrow$ *kräfte*, and *kampf*$\rightarrow$ *kämpfe*. The smallest classes were singletons, with highly irregular plurals: *atlas/atlanten* and *opus/opera* being good examples. These results are not perfect; some words that take the same type of plural end up in different classes.

Table 2: The largest seven models from the German plural data set. Models 4 and 7 represent the same transduction, namely `+en`

i	States	Words	Stem	Plural
0	8	699	Winter	Winter
1	9	736	Wille	Willen
2	9	715	Werk	Werke
3	9	199	Papa	Papas
4	11	1285	Welt	Welten
5	12	128	Nacht	Nächte
7	12	85	Pfau	Pfauen

One of the forms of the Arabic plural(McCarthy and Prince, 1990), the *broken* plural is a highly nonconcatenative process that maps for example *bank* to *bunuuk*. There are various forms of this that are sensitive to the prosodic outline of the stem. Preliminary experiments with PHMMs confirmed that they were capable of learning the individual transductions involved and to a more limited extent modelling the prosodic features that determine their applicability. Training on a data set prepared by Nakisa and Plunkett(1997) produced 55 separate models which in general corresponded to various specific forms of the plural. However this data is rather sparse, and overly simplified.

9 Discussion

There is a large amount of literature on learning the English past tense, (Rumelhart and McLelland, 1986; Ling, 1994; Mooney and Califf, 1995). There is much less work on more morphologically complex languages (Plunkett and Nakisa, 1997). PHMMs are close to the Stochastic Inversion Transduction Grammars of Wu (Wu, 1997), and also to Ristad's memoryless string transducers(Ristad and Yianilos, 1998). They are also related to the asynchronous IOHMMs of (Bengio and Frasconi, 1996). They can be thought of as nondeterministic stochastic finite-state transducers.

Though PHMMs can model a wide range of morphophonological phenomena, they cannot model reduplication, a phenomenon found in many languages such as Malay and Tagalog. If however, we considered a slightly richer model which outputs 3 streams of symbols, where two of the streams are copies, it should be possible to cover these languages as well.

German nouns have eight possible morphosyntactic forms – singular and plural with four cases each, though there are at most four distinct forms. It would be interesting to model the full inflectional system simultaneously using a set of seven or eight PHMMs for each class. This would allow a more thorough evaluation of the generalisation properties of the algorithm.

It is possible to use these models for the *unsupervised* acquisition of morphology, using the EM algorithm again to model the alignments between words.

In this paper I have presented a novel algorithm for learning non-deterministic stochastic transductions, together with an application to the acquisition of morphology. This algorithm can learn various morphological processes from a range of languages with a high degree of accuracy. It does this without using any specifically linguistic knowledge. It does not suffer from the limitations of previous approaches such as *ad hoc* coding schemes, limitations on the length of input and output strings, or hard-coded preferences for particular transductions.

Acknowledgments

I am grateful to Bill Keller and Gerald Gazdar for critical discussions. This work was partly done for the European TMR Network *Learning Computational Grammars* at the University of Geneva.

References

L. E. Baum and T. Petrie. 1966. Statistical inference for probabilistic functions of finite state markov chains. *Annals of Mathematical Statistics*, 37:1559–1663.

Y. Bengio and P. Frasconi. 1996. Input/output HMMs for sequence processing. *IEEE Transactions on Neural Networks*, 7(5).

L. J. Cahill and G. Gazdar. 1999. German noun inflection. *Journal of Linguistics*, 35(1):1–42.

Francisco Casacuberta and Colin de la Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. In Arlindo L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications*, number 1891 in Lecture Notes in Artificial Intelligence, pages 15–24. Springer Verlag.

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

Ronald M. Kaplan and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378, September.

Kimmo Koskenniemi. 1983. *A Two-level Morphological Processor.* Ph.D. thesis, University of Helsinki.

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.

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

E. S. Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

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.

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