N-th Order Ergodic Multigram HMM for Modeling of Languages without Marked Word Boundaries

Hubert Hin-Cheung LAW
Dept. of Computer Science
The University of Hong Kong
hhclaw@cs.hku.hk

Chorkin CHAN
Dept. of Computer Science
The University of Hong Kong
cchan@cs.hku.hk

Abstract

Ergodic HMMs have been successfully used for modeling sentence production. However for some oriental languages such as Chinese, a word can consist of multiple characters without word boundary markers between adjacent words in a sentence. This makes word-segmentation on the training and testing data necessary before ergodic HMM can be applied as the language model. This paper introduces the N-th order Ergodic Multigram HMM for language modeling of such languages. Each state of the HMM can generate a variable number of characters corresponding to one word. The model can be trained without word-segmented and tagged corpus, and both segmentation and tagging are trained in one single model. Results on its application on a Chinese corpus are reported.

1 Motivation

Statistical language modeling offers advantages including minimal domain specific knowledge and hand-written rules, trainability and scalability given a language corpus. Language models, such as N-gram class models (Brown et al., 1992) and Ergodic Hidden Markov Models (Kuhn et al., 1994) were proposed and used in applications such as syntactic class (POS) tagging for English (Cutting et al., 1992), clustering and scoring of recognizer sentence hypotheses.

However, in Chinese and many other oriental languages, there are no boundary markers, such as space, between words. Therefore preprocessors have to be used to perform word segmentation in order to identify individual words *before* applying these word-based language models. As a result current approaches to modeling these languages are separated into two seperated processes.

Word segmentation is by no means a trivial process, since ambiguity often exists. For proper segmentation of a sentence, some linguistic information of the sentence should be used. However, commonly used heuristics or statistical based approaches, such as maximal matching, frequency counts or mutual information statistics, have to perform the segmentation without knowledge such as the resulting word categories.

To reduce the impact of erroneous segmentation on the subsequent language model, (Chang and Chan, 1993) used an N-best segmentation interface between them. However, since this is still a two stage model, the parameters of the whole model cannot be optimized together, and an N-best interface is inadequate for processing outputs from recognizers which can be highly ambiguous.

A better approach is to keep all possible segmentations in a lattice form, score the lattice with a language model, and finally retrieve the best candidate by dynamic programming or some searching algorithms. N-gram models are usually used for scoring (Gu et al., 1991) (Nagata, 1994), but their training requires the sentences of the corpus to be manually segmented, and even class-tagged if class-based N-gram is used, as in (Nagata, 1994).

A language model which considers segmentation ambiguities and integrates this with a N-gram model, and able to be trained and tested on a raw, unsegmented and untagged corpus, is highly desirable for processing languages without marked word boundaries.

2 The Ergodic Multigram HMM Model

2.1 Overview

Based on the Hidden Markov Model, the Ergodic Multigram Hidden Markov Model (Law and Chan, 1996), when applied as a language model, can process directly on unsegmented input corpus

as it allows a variable number of characters in each word class. Other than that its properties are similar to Ergodic Hidden Markov Models (Kuhn et al., 1994), that both training and scoring can be done directly on a raw, untagged corpus, given a lexicon with word classes.

Specifically, the N-th order Ergodic Multigram HMM, as in conventional class-based $(N+1)$-gram model, assumes a doubly stochastic process in sentence production. The word-class sequence in a sentence follows the N-th order Markov assumption, i.e. the identity of a class in the sentence depends only on the previous N classes, and the word observed depends only on the class it belongs to. The difference is that this is a multigram model (Deligne and Bimbot, 1995) in the sense that each state (i.e. node in the HMM) can generate a variable number of observed character sequences. Sentence boundaries are modeled as a special class.

This model can be applied to an input sentence or a character lattice as a language model. The maximum likelihood state sequence through the model, obtained using the Viterbi or Stack Decoding Algorithm, represents the best particular segmentation and class-tagging for the input sentence or lattice, since transition of states denotes a word boundary and state identity denotes the current word class.

2.2 Lexicon

A lexicon (CKIP, 1993) of 78,322 words, each containing up to 10 characters, is available for use in this work. Practically all characters have an entry in the lexicon, so that out-of-vocabulary words are modeled as individual characters. There is a total of 192 syntactic classes, arranged in a hierarchical way. For example, the month names are denoted by the class `Ndabc`, where `N` denotes Noun, `Nd` denotes Temporal Nouns, `Nda` for Time names and `Ndab` for reusable time names. There is a total of 8 major categories.

Each word in the dictionary is annotated with one or more syntactic tags, representing different syntactic classes the word can possibly belong to. Also, a frequency count for each word, based on a certain corpus, is given, but without information on its distribution over different syntactic classes.

2.3 Terminology

Let $\mathcal{W}$ be the set of all Chinese words in the lexicon. A word $w_k \in \mathcal{W}$ is made up of one or more characters. Let $s_1^T = (s_1, s_2, \ldots s_T)$ denote a sentence as a T-character sequence. A function δ_w is defined such that $\delta_w(w_k, s_t^{t+r-1})$ is 1 if w_k is a r-character word $s_t \ldots s_{t+r-1}$, and 0 otherwise. [1] Let R be the upper bound of r, i.e. the maximum number of characters in a word (10 in this paper).

Let $C = \{c_1 \ldots c_L\}$ be the set of syntactic classes, where L is the number of syntactic classes in the lexicon (192 in our case). Let $\mathcal{C} \subset \mathcal{W} \times C$ denote the relation for all syntactic classifications of the lexicon, such that $(w_k, c_l) \in \mathcal{C}$ iff c_l is one of the syntactic classes for w_k. Each word w_k must belong to one or more of the classes.

A path through the model represents a particular segmentation and class tagging for the sentence. Let $\mathcal{L} = (w_1, c_{l_1}; \ldots; w_K, c_{l_K})$ be a particular segmentation and class tagging for the sentence s_1^T, where w_k is the kth word and c_{l_k} denotes the class assigned to w_k, as illustrated below.

$$(\overbrace{s_1 \ldots s_{t_1-1}}^{w_1, c_{l_1}} \ldots \overbrace{s_{t_{k-1}} \ldots s_{t_k-1}}^{w_k, c_{l_k}} \ldots \overbrace{s_{t_{K-1}} \ldots s_T}^{w_K, c_{l_K}})$$

For $\mathcal{L}$ to be proper, $\prod_{k=1}^{K} \delta_w(w_k, s_{t_{k-1}}^{t_k-1}) = 1$ and $(w_k, c_{l_k}) \in \mathcal{C}$ must be satisfied, where $t_0 = 1, t_K = T+1$ and $t_{k-1} < t_k$ for $1 \le k \le K$.

2.4 HMM States for the N-th order model

In the first order HMM (class bigram) model, each HMM state corresponds directly to the word-class of a word. But in general, for an N-th order HMM model, since each class depends on N previous classes, each state has to represent the combination of the classes of the most recent N words, including the current word.

Let Q_i represent a state of the N-th order Ergodic Multigram HMM. Thus $Q_i = (c_{i_0} \ldots c_{i_{N-1}})$ where c_{i_0} is the current word class, c_{i_1} is the previous word class, etc. There is a total of L^N states, which may mean too many parameters (L^{N+1} possible state transitions, each state can transit to L other states) for the model if N is anything greater than one.

To solve this problem, a reasonable assumption can be made that the detailed class identities of a more distant word have, in general, less influence than the closer ones to the current word class. Thus instead of using $\mathcal{C}$ as the classification relation for all previous words, a set of

[1] The algorithm to be described assumes that the character identities are known for the sentence s_1^T, but it can also be applied when each character position s_t becomes a set of possible character candidates by simply letting $\delta_w(w_k, s_t^{t+r-1}) = 1$ for all words w_k which can be constructed from the character positions $s_t \ldots s_{t+r-1}$ of the input character lattice. This enables the model to be used as the language model component for recognizers and for decoding phonetic input.

classification relations $\{\mathcal{C}^{(0)}, \mathcal{C}^{(1)}, \ldots \mathcal{C}^{(N-1)}\}$ can be used, where $\mathcal{C}^{(0)} = \mathcal{C}$ represents the original, most detailed classification relation for the current word, and $\mathcal{C}^{(n)}$ is the less detailed classification scheme for the nth previous word at each state. Thus the number of states reduces to $L_Q = L^{(0)}L^{(1)} \ldots L^{(N-1)}$ in which $L^{(n)} \leq L$. Each state is represented as $Q_i = (c_{i_0}^{(0)} \ldots c_{i_{N-1}}^{(N-1)})$ where $C^{(n)} = \{c_l^{(n)}\}$, $1 \leq l \leq L^{(n)}$ is the class tag set for the nth previous word.

However, if no constraints are imposed on the series of classification relations $\mathcal{C}^{(n)}$, the number of possible transitions may increase despite a decrease in the number of states, since state transitions may become possible between every two state, resulting in a total of ${L^{(0)}}^2{L^{(1)}}^2 \ldots {L^{(N-1)}}^2$ possible transitions.

A constraint is imposed that, given that a word belongs to the class $c_l^{(n)}$ in the classification $\mathcal{C}^{(n)}$, we can determine the corresponding word class $c_{l'}^{(n+1)}$ the given word will belong to in $\mathcal{C}^{(n+1)}$, and for every word there is no extra classifications in $\mathcal{C}^{(n+1)}$ not corresponding to one in $\mathcal{C}^{(n)}$. Formally, there exist mapping functions $\mathcal{F}^{(n)} : C^{(n)} \mapsto C^{(n+1)}$, $0 \leq n \leq N-2$, such that if $(c_l^{(n)}, c_{l'}^{(n+1)}) \in \mathcal{F}^{(n)}$, then $((w_k, c_l^{(n)}) \in \mathcal{C}^{(n)}) \Rightarrow ((w_k, c_{l'}^{(n+1)}) \in \mathcal{C}^{(n+1)})$ for all $w_k \in \mathcal{W}$, and that $\mathcal{F}^{(n)}$ is surjective. In particular, to model sentence boundaries, we allow \$ to be a valid class tag for all $C^{(n)}$, and define $\mathcal{F}^{(n)}(\$) = \$$.

The above constraint ensures that given a state

$$Q_i = (c_{i_0}^{(0)} \ldots c_{i_{N-1}}^{(N-1)})$$

it can only transit to

$$Q_j = (c_{j_0}^{(0)}, \mathcal{F}^{(0)}(c_{i_0}^{(0)}) \ldots \mathcal{F}^{(N-2)}(c_{i_{N-2}}^{(N-2)}))$$

where $c_{j_0}^{(0)}$ is any state in $C^{(0)}$. Thus reducing to the maximum number of possible transitions to ${L^{(0)}}^2 L^{(1)} \ldots L^{(N-1)}$.

This constraint is easily satisfied by using a hierarchical word-class scheme, such as the one in the CKIP lexicon or one generated by hierarchical word-clustering, so that the classification for more distant words (higher n in $\mathcal{C}^{(n)}$) uses a higher level, less detail tag set in the scheme.

2.5 Sentence Likelihood Formulation

Let $\{\mathcal{L}\}$ be the set of all possible segmentations and class taggings of a sentence. Under the N-th order model Θ^N, the likelihood of each valid segmentation and tagging $\mathcal{L}$ of the sentence s_1^T, $P(s_1^T, \mathcal{L}|\Theta^N)$, can be derived as follows.

$$\begin{aligned} & P(w_1, c_{l_1}; w_2, c_{l_2}; \ldots; w_K, c_{l_K}|\Theta^N) \\ = \; & P(w_1|c_{l_1})P(c_{l_1}|\$^N)P(\$|c_{l_K} \ldots c_{l_{K-N+1}}) \times \\ & (\textstyle\prod_{k=2}^{K} P(w_k|c_{l_k})P(c_{l_k}|c_{l_{k-1}} \ldots c_{l_{k-N}})) \\ = \; & P(w_1|c_{l_1})P(Q_{l_1}|\$^N)P(\$|Q_{l_K}) \times \\ & (\textstyle\prod_{k=2}^{K} P(w_k|c_{l_k})P(Q_{l_k}|Q_{l_{k-1}})) \end{aligned}$$

using Nth order Markov assumption and representing the class history as HMM states. \$ denotes the sentence boundary, c_{l_k} is \$ for $k \leq 0$, and $Q_{l_k} = (c_{l_k}^{(0)} \ldots c_{l_{k-N+1}}^{(N-1)})$. Note that Q_{l_k} can be determined from c_{l_k} and $Q_{l_{k-1}}$ due to the constraint on the classification, and thus $P(Q_{l_k}|Q_{l_{k-1}}) = P(c_{l_k}|Q_{l_{k-1}})$.

The likelihood of the sentence s_1^T under the model is given by the sum of the likelihoods of its possible segmentations.

$$P(s_1^T|\Theta^N) = \sum_{\mathcal{L} \in \{\mathcal{L}\}} P(s_1^T, \mathcal{L}|\Theta^N)$$

3 The Algorithms

3.1 The Parameters

As in conventional HMM, the Ergodic Multigram HMM consists of parameters $\Theta^N = \{A, B\}$, in which $A = \{a_{ij}\}$, $0 \leq i, j \leq L_Q$ (Total number of states), denotes the set of state transition probabilities from Q_i to Q_j, i.e. $P(Q_j|Q_i)$. In particular, $a_{0i} = P(Q_i|\$^N)$ and $a_{i0} = P(\$|Q_i)$ denote the probabilities that the state Q_i is the initial and final state in traversing the HMM, respectively. a_{00} is left undefined. $B = \{b_j(w_k)\}$, where $1 \leq j \leq L^Q$, denotes the set of word observation probabilities of w_k at the state Q_j, i.e. $P(w_k|Q_j)$.

The B matrix, as shown above, models the probabilities that w_k is observed given N most recent classes, and contains $L_Q|\mathcal{W}|$ parameters (recall that $L_Q = L^{(0)}L^{(1)} \ldots L^{(N-1)}$). Our assumption that w_k only depends on the current class reduces the number of parameters to $L^{(0)}|\mathcal{W}|$ for the B matrix. Thus in the model, $b_j(w_k)$ representing $P(w_k|Q_j)$ are tied together for all states Q_j with the same current word-class, i.e. $P(w_k|Q_j) = P(w_k|c_l)$ if $Q_j = (c_l \ldots)$. Also, a_{ij} is 0 if Q_i cannot transit to Q_j. As a result the number of parameters in the A matrix is only $L^{(0)}L_Q$.

Given the segmentation and class sequence $\mathcal{L}$ of a sentence, the state sequence $(Q_{l_1} \ldots Q_{l_k})$ can be derived from the class sequence $(c_{l_1} \ldots c_{l_k})$. Thus the observation probability of the sentence s_1^T given $\mathcal{L}$ and the model Θ^N, $P(s_1^T, \mathcal{L}|\Theta^N)$, can be reformulated as

$$b_{l_1}(w_1)a_{0l_1}a_{l_K 0}(\textstyle\prod_{k=2}^{K} a_{l_{k-1}l_k}b_{l_k}(w_k))$$

Given this formulation the training procedure is mostly similar to that of the first order Ergodic Multigram HMM.

3.2 Forward and Backward Procedure

The forward variable is defined as

$$\alpha_t(i) = P(s_1 \dots s_t, Q_{l(t)} = Q_i | \Theta^N)$$

where $Q_{l(t)}$ is the state of the HMM when the word containing the character s_t *as the last character* is produced.

The recursive equations for $\alpha_t(i)$ are

$$\begin{aligned}
\alpha_t(j) &= 0 \text{ for } t < 1 \\
\alpha_t(j) &= \sum_{w_k \in \mathcal{W}} a_{0j} b_j(w_k) \delta_w(w_k, s_1^t) + \\
&\quad \sum_{r=1}^{R} \sum_{w_k \in \mathcal{W}} [\sum_{i=1}^{L_Q} \alpha_{t-r}(i) a_{ij} b_j(w_k)] \\
&\quad \delta_w(w_k, s_{t-r+1}^t) \\
&\quad \text{for } 1 \le t \le T
\end{aligned}$$

Similarly, the backward variable is defined as

$$\beta_t(i) = P(s_{t+1} \dots s_T | Q_{l(t)} = Q_i, \Theta^N)$$

The recursive equations for $\beta_t(i)$ are

$$\begin{aligned}
\beta_t(i) &= 0 \text{ for } t > T \\
\beta_T(i) &= a_{i0} \\
\beta_t(i) &= \sum_{r=1}^{R} \sum_{w_k \in \mathcal{W}} [\sum_{j=1}^{L_Q} a_{ij} \beta_{t+r}(j) b_j(w_k)] \\
&\quad \delta_w(w_k, s_{t+1}^{t+r}) \\
&\quad \text{for } 1 \le t \le T-1
\end{aligned}$$

As A, B arrays and the δ_w function are mostly 0s, considerable simplification can be done in implementation.

The likelihood of the sentence given the model can be evaluated as

$$P(s_1^T | \Theta^N) = \sum_{i=1}^{L_Q} \alpha_T(i) a_{i0}$$

The Viterbi algorithm for this model can be obtained by replacing the summations of the forward algorithm with maximizations.

3.3 Re-estimation Algorithm

$\xi_t(i,j)$ is defined as the probability that given a sentence s_1^T and the model Θ^N, a word ends at the character s_t in the state Q_i and the next word starts at the character s_{t+1} in the state Q_j. Thus $\xi_t(i,j)$ can be expressed as

$$\frac{\sum_{r=1}^{R} \sum_{w_k \in \mathcal{W}} \alpha_t(i) a_{ij} b_j(w_k) \delta_w(w_k, s_{t+1}^{t+r}) \beta_{t+r}(j)}{P(s_1^T | \Theta^N)}$$

for $1 \le t \le T-1, 1 \le i,j \le L_Q$. Furthermore define $\gamma_t(i)$ to be the probability that, given s_1^T and Θ^N, a word ends at the character s_t in the state Q_i. Thus

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(s_1^T | \Theta^N)} \text{ for } 1 \le t \le T, 1 \le i \le L_Q.$$

Summation of $\xi_t(i,j)$ over t gives the expected number of times that state Q_i transits to state Q_j in the sentence, and summation of $\gamma_t(i)$ over t gives the expected number of state Q_i occurring in it. Thus the quotient of their summation over t gives $\bar{a}_{ij}$, the new estimation for a_{ij}.

$$\bar{a}_{ij} = \sum_{t=1}^{T-1} \xi_t(i,j) / \sum_{t=1}^{T} \gamma_t(i) \text{ for } 1 \le i,j \le L_Q$$

The initial and final class probability estimates, $\bar{a}_{0j}$ and $\bar{a}_{i0}$ can be re-estimated as follows.

$$\begin{aligned}
\bar{a}_{0j} &= \frac{\sum_{r=1}^{R} \sum_{w_k \in \mathcal{W}} a_{0j} b_j(w_k) \delta_w(w_k, s_1^r) \beta_r(j)}{P(s_1^T | \Theta^N)} \\
\bar{a}_{i0} &= \frac{\alpha_T(i) a_{i0}}{P(s_1^T | \Theta^N)} / \sum_{t=1}^{T} \gamma_t(i)
\end{aligned}$$

To derive $b_j(w_k)$, first define $\alpha_t^{w_k}(i)$ as the probability of the sentence prefix $(s_1 \dots s_t)$ with w_k in state Q_i as the last complete word. Thus

$$\begin{aligned}
\alpha_t^{w_k}(j) &= a_{0j} b_j(w_k) \delta_w(w_k, s_1^t) + \sum_{r=1}^{R} \sum_{i=1}^{L_Q} \\
&\quad (\alpha_{t-r}(i) a_{ij} b_j(w_k) \delta_w(w_k, s_{t-r+1}^t))
\end{aligned}$$

This represents the contribution of w_k, occurring as the last word in s_1^t, to $\alpha_t(j)$. Also define $\gamma_t^{w_k}(j)$ to be the probability that, given the sentence s_1^T and the model, w_k is observed to end at character s_t in the state Q_j.

$$\gamma_t^{w_k}(j) = \frac{\alpha_t^{w_k}(j)\beta_t(j)}{P(s_1^T | \Theta^N)}$$

Let $Q_j \diamond Q_{j'}$ denotes the relation that both Q_j and $Q_{j'}$ represent the same current word class. Thus summation of $\gamma_t^{w_k}(j)$ over t gives the expected number of times that w_k is observed in

state Q_j, and summation of $\gamma_t(j)$ over t gives the total expected number of occurrence of state Q_j. Since states with the same current word class are tied together by our assumption, the required value of $\bar{b}_j(w_k)$ is given by

$$\bar{b}_j(w_k) = \frac{\sum_{\substack{j' \\ Q_j \diamond Q_{j'}}} \sum_{t=1}^{T} \gamma_t^{w_k}(j')}{\sum_{\substack{j' \\ Q_j \diamond Q_{j'}}} \sum_{t=1}^{T} \gamma_t(j')}$$

4 Experimental Results

4.1 Setup

A corpus of daily newspaper articles is divided into training and testing sets for the experiments, which is 21M and 4M in size respectively. The first order (N=1) algorithms are applied to the training sets, and parameters obtained after different iterations are used for testing.

The initial parameters of the HMM are set based on the frequency counts from the lexicon. The class-transition probability a_{ij} is initialized as the a priori probability of the state $P(Q_j)$, estimated from the relative frequency counts of the lexicon. $b_j(w_k)$ is initialized as the relative count of the word w_k within the class corresponding to the current word class in Q_j. Words belonging to multiple classes have their counts distributed equally among them. Smoothing is then applied by adding each word count by 0.5 and normalizing.

After training, the Viterbi algorithm is used to retrieve the best segmentation and tagging $\mathcal{L}^*$ of each sentence of the test corpus, by tracing the best state sequence traversed.

4.2 Perplexity

The test-set perplexity, calculated as

$$PP^* = \exp(-\frac{1}{M}\sum_i \log(P(s_1^{T_i}, \mathcal{L}^*|\Theta^N)))$$

where the summation is taken over all sentences $s_1^{T_i}$ in the testing corpus, and M represents the number of *characters* in it, is used to measure the performance of the model.

The results for models trained on training corpus subsets of various sizes, and after various iterations are shown (Table 1). It is obvious that with small training corpus, over-training occurs with more iterations. With more training data, the performance improves and over-training is not evident.

4.3 Phonetic Input Decoding

A further experiment is performed to use the models to decode phonetic inputs (Gu et al., 1991).

Training Size	2	4	6	8
98K	194.009	214.096	246.613	286.721
1.3M	126.084	122.304	121.606	121.776
6.3M	118.531	113.600	111.745	110.783
21M	116.376	111.275	109.282	108.142

Table 1: Test Set Perplexities of testing set after different iterations on subsets of training set

This is not trivial since each Chinese syllable can correspond to up to 80 different characters. Sentences from the testing corpus are first expanded into a lattice, formed by generating all the common homophones of each Chinese character. Tested on 360K characters, a character recognition rate of 91.24% is obtained for the model trained after 8 iterations with 21M of training text. The results are satisfactory given that the test corpus contains many personal names and out of vocabulary words, and the highly ambiguous nature of the problem.

5 Discussion and Conclusion

In this paper the N-th order Ergodic Multigram HMM is introduced, whose application enables integrated, iterative language model training on untagged and unsegmented corpus in languages such as Chinese.

The performance on higher order models are expected to be better as the size of training corpus is relatively large. However some form of smoothing may have to be applied when the training corpus size is small.

With some modification this algorithm would work on phoneme candidate input instead of character candidate input. This is useful in decoding phonetic strings without character boundaries, such as in continuous Chinese/Japanese/Korean phonetic input, or speech recognizers which output phonemes.

This model also makes a wealth of techniques developed for HMM in the speech recognition field available for language modeling in these languages.

References

Brown, P.F., deSouza, P.V., Mercer, R.L., Della Pietra, V.J., Lai, J.C. 1992. Class-Based n-gram Models of Natural Language. In *Computational Linguistics*, 18:467-479.

Chang, C.H., Chan, C.D. 1993. A Study on Integrating Chinese Word Segmentation and Part-

of-Speech Tagging. In *Comm. of COLIPS, Vol 3, No.1*, pp.69-77.

Chinese Knowledge Information Group 1993. In *Technical Report No. 93-05*. Institute of Information Science, Academia Sinica, Taiwan.

Cutting, K., Kupiec, J., Pedersen, J., Sibun, P. 1992. A Practical Part-of-Speech Tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pp.133-140.

Deligne, S., Bimbot, F. 1995. Language Modeling by Variable Length Sequences: Theoretical Formulation and Evaluation of Multigrams. In *ICASSP 95*, pp.169-172.

Gu, H.Y., Tseng, C.Y., Lee, L.S. 1991. Markov Modeling of Mandarin Chinese for decoding the phonetic sequence into Chinese characters. In *Computer Speech and Language, Vol 5*, pp.363-377.

Kuhn, T., Niemann, H., Schukat - Talamazzini, E.G. 1994. Ergodic Hidden Markov Models and Polygrams for Language Modeling. In *ICASSP 94*, pp.357-360.

Law, H.H.C., Chan, C. 1996. Ergodic Multigram HMM Integrating Word Segmentation and Class Tagging for Chinese Language Modeling. To appear in *ICASSP 96*.

Nagata, M. 1994. A Stochastic Japanese Morphological Analyzer Using a Forward-DP Backward-A* N-Best Search Algorithm. In *COLING 94*, pp.201-207.