# ERGODIC MULTIGRAM HMM INTEGRATING WORD SEGMENTATION AND CLASS TAGGING FOR CHINESE LANGUAGE MODELING

*Hubert Hin-Cheung LAW and Chorkin CHAN*

Department of Computer Science
The University of Hong Kong
email: {hhclaw,cchan}@cs.hku.hk

## ABSTRACT

A novel Ergodic Multigram Hidden Markov Model (HMM) is introduced which models sentence production as a doubly stochastic process, in which word classes are first produced according to a first order Markov model, and then single or multi-character words are generated independently based on the word classes, *without* word boundary marked on the sentence.

This model can be applied to languages without word boundary markers such as Chinese. With a lexicon containing syntactic classes for each word, its applications include language modeling for recognizers, and integrated word segmentation and class tagging. Pre-segmented and tagged corpus are not needed for training, and both segmentation and tagging are trained in one single model. In this paper, relevant algorithms for this model are presented, and experimental results on a Chinese news corpus are reported.

## 1. INTRODUCTION

Language modeling was shown capable of improving recognizer performance. Statistical models including $N$-gram class models [1] and Ergodic Hidden Markov Models [2] were proposed. However, in languages such as Chinese, where there are no boundary markers between words, *word segmentation* is needed to identify individual words before applying these word-based models and essentially any further processing. Popular methods for word segmentation include maximal matching, frequency counts, mutual information statistics, and rule based heuristics.

*Syntactic class tagging* of words are often useful for both language modeling and further processing of a sentence. Various tagging methods were proposed for English [3] [4]. Again, they can only be applied to Chinese after word segmentation.

However, decision on segmentation without considering higher level linguistic information, such as syntactic classes of the resulting words, may be error-prone. A better approach is to first produce the N-best segmentations of a sentence based on segmentation scores, and then apply a language model such as HMM to further score each of them. The final score of a candidate is the weighted sum of its two scores [5]. However, parameters of this two stage model cannot be trained and optimized together, and an N-best interface is inadequate for processing highly ambiguous character lattices from the recognizer output.

Another 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 [6] [7], but their training requires the sentences of the corpus to be segmented (and tagged if class-based $N$-gram is used [7]).

We introduce the Ergodic Multigram Hidden Markov Model which, when applied as a language model for these languages, integrates the segmentation and tagging processes into one model, and does not assume any prior segmentation or class tagging. Thus both training and scoring can be done using the model directly on a raw corpus. This model can be applied as a language model for an input sentence (or a character lattice), and the maximum likelihood segmentation and class-tagging of it can be obtained using the Viterbi or Stack Decoding Algorithm.

## 2. 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 $s_1^T = (s_1, s_2, \ldots s_T)$ denotes a sentence as a $T$-character sequence. A function $\delta_w$ can be define

$$\delta_w(w_k, s_t^{t+r-1}) = \begin{cases} 1 & \text{if } w_k \text{ is a } r\text{-character word} \\ & s_t \ldots s_{t+r-1} \\ 0 & \text{otherwise} \end{cases}$$

Let $R$ be the allowed upper bound of $r$, i.e. the maximum number of characters in a word. Let $L$ be the number of syntactic classes in the lexicon, and $C = \{c_1 \ldots c_L\}$ the set of all syntactic classes. Each word $w_k$ belongs to one or more of these classes.

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 $k$th word and $c_{l_k}$ denotes the class assigned to $w_k$, as illustrated below.

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

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

The first condition ensures that $\mathcal{L}$ is a proper segmentation of the sentence into words, and the second ensures valid class tagging of the words.

0-7803-3192-3/96 $5.00©1996 IEEE

## 3. THE LANGUAGE MODEL

An Ergodic Multigram [8] Hidden Markov Model is constructed to model a source of sentences. We assume that the class of a word in a sentence only depends on the previous class, and the word observed in turn only depends on the class it belongs to. This is similar to the assumptions of the conventional word-based ergodic HMM [2], but the proposed model is based on characters observed rather than word units. In other words it is a multigram HMM in which every state can generate a variable number of observed character sequences. As in most bigram modeling, sentence boundary is modeled as a special class.

The model consists of parameters $\Theta = \{A, B\}$, where $A = \{a_{ij}\}$ is the set of word-class transition probabilities $P(c_j|c_i)$, and $B = \{b_j(w_k)\}$ is the set of word observation probabilities $P(w_k|c_j)$.

$a_{ij}$, where $1 \leq i, j \leq L$, denotes the probability that the word class is $c_j$ given the previous word class in the sentence is $c_i$, while $a_{0i} = P(c_i|\$)$ and $a_{i0} = P(\$|c_i)$, where $1 \leq i \leq L$ and \$ denotes the sentence boundary, denote the probabilities that the class $c_i$ is the first and last word class in the sentence, respectively. $a_{00}$ is left undefined. $b_j(w_k)$, where $1 \leq j \leq L$, denotes the probability that the word observed is $w_k$ given the class is $c_j$.

Let $\{\mathcal{L}\}$ be the set of all possible segmentations and class taggings of a sentence. For a valid segmentation $\mathcal{L}$ of the sentence $s_1^T$, given the model $\Theta$, its likelihood is given by

$$\begin{aligned} P(s_1^T, \mathcal{L}|\Theta) &= P(w_1, c_{l_1}; w_2, c_{l_2}; \ldots; w_K, c_{l_K}|\Theta) \\ &= P(w_1|c_{l_1})P(c_{l_1}|\$)P(\$|c_{l_K}) \times \\ &\quad (\prod_{k=2}^{K} P(w_k|c_{l_k})P(c_{l_k}|c_{l_{k-1}})) \\ &= a_{0l_1} b_{l_1}(w_1) a_{l_K 0} (\prod_{k=2}^{K} a_{l_{k-1} l_k} b_{l_k}(w_k)) \end{aligned}$$

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

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

## 4. THE ALGORITHMS

### 4.1. Forward and Backward Procedure

The forward variable is defined as

$$\alpha_t(i) = P(s_1 \ldots s_t, c_{l(t)} = c_i|\Theta)$$

where $c_{l(t)}$ is the class assigned to the word containing the character $s_t$ *as the last character.* i.e. only segmentations up to the word boundary just after $s_t$ contribute to this probability. This denotes, given the model $\Theta$, the likelihood of the sentence prefix $(s_1 \ldots s_t)$ with the last complete word of class $c_i$.

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

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

Similarly, the backward variable is defined as

$$\beta_t(i) = P(s_{t+1} \ldots s_T|c_{l(t)} = c_i, \Theta)$$

Again only segmentations with word boundaries up to the one just before $s_{t+1}$ (i.e. after $s_t$) are counted. This denotes the likelihood of the sentence suffix $(s_{t+1} \ldots s_T)$ given the model and that the immediately preceding complete word is of class $c_i$.

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

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

The time complexity of the above procedures are not so much as $O(TL^2R|\mathcal{W}|)$ as it may seem, since $\delta_w(w_k, s_{t_1}^{t_2}) = 0$ for all but one $w_k$, with $r = t_2 - t_1 + 1$ known for this $w_k$. So for actual implementation an enumeration of all words starting or ending at a given character position is used, instead of expensive loops over all $1 \leq r \leq R$ and $w_k \in \mathcal{W}$. Also $b_j(w_k)$ is non-zero only if $w_k \in c_j$.

Thus let $L'$ be the maximum number of classes a word can belong to, and $N$ be the maximum number of words beginning or ending at any character position. The time complexity is just $O(TLL'N)$.

The likelihood of the sentence given the model can be evaluated by both the forward and backward variables.

$$\begin{aligned} P(s_1^T|\Theta) &= \sum_{i=1}^{L} \alpha_T(i) a_{i0} \\ &= \sum_{r=1}^{R} \sum_{w_k \in \mathcal{W}} [\sum_{j=1}^{L} a_{0j} \beta_r(j) b_j(w_k)] \delta_w(w_k, s_1^r) \end{aligned}$$

### 4.2. Viterbi Algorithm

The most likely segmentation and class assignment $\mathcal{L}^*$ of a given sentence can be retrieved using a Viterbi-like algorithm, obtained by replacing the summations of the forward algorithm with maximizations. Top $N$ segmentation and class assignment candidates are similarly retrieved using an A* algorithm, similar to the word lattice searching algorithm based on dynamic programming [6].

### 4.3. Re-estimation Algorithm

$\xi_t(i,j)$ is defined as the probability that given a sentence $s_1^T$ and the model $\Theta$, a word of class $c_i$ ends at the character $s_t$ and a word of class $c_j$ starts at the character $s_{t+1}$. Thus

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

for $1 \le t \le T-1, 1 \le i,j \le L$. Furthermore define $\gamma_t(i)$ to be the probability that, given $s_1^T$ and $\Theta$, word class $c_i$ ends at the character $s_t$. Thus

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

The quotient of their summation over $t$ gives $\bar{a}_{ij}$, the new estimation for $a_{ij}$.

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

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

$$\bar{a}_{0j} = \frac{\sum_{r=1}^{R} \sum_{w_k \in \mathcal{W}} a_{0j} b_j(w_k) \delta_w(w_k, s_1^r) \beta_r(j)}{P(s_1^T|\Theta)}$$

$$\bar{a}_{i0} = \frac{\alpha_T(i) a_{i0}}{P(s_1^T|\Theta)} / \sum_{t=1}^{T} \gamma_t(i)$$

To derive $\bar{b}_j(w_k)$, first define $\alpha_t^{w_k}(i)$ as the probability of the sentence prefix $(s_1 \ldots s_t)$ with $w_k$ of class $c_i$ as the last complete word, given the model $\Theta$. Thus

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

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 $s_t$ and assigned class $c_j$.

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

The required value of $\bar{b}_j(w_k)$ is then given by:

$$\bar{b}_j(w_k) = \sum_{t=1}^{T} \gamma_t^{w_k}(j) / \sum_{t=1}^{T} \gamma_t(j)$$

This algorithm assumes that the identities of the Chinese characters are known for the sentence $s_1^T$, but it can equally well be applied to recognizer generated character lattices or phonetic input, where 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 input character lattice. This enables the model to be used as the language model component for recognizers and for decoding phonetic input.

## 5. EXPERIMENTAL RESULTS

For evaluation, the training algorithm above is implemented with simple smoothing. A variant training algorithm, whose re-estimation bases on relative frequency (RF) counts from the Viterbi segmentation and tagging $\mathcal{L}^*$, is also implemented with back-off smoothing [9].

A lexicon [10] of 78,322 words and 192 syntactic classes ($L = 192$), with the maximum word length of 10 characters ($R = 10$) is used. Each word entry is associated with its class tags and a frequency count.

The initial parameters of the HMM are based on the frequency counts from the lexicon. The class-transition probability $a_{ij}$ is initialized as the a priori probability of the class $P(c_j)$, and $b_j(w_k)$ as the relative count of the word $w_k$ within the class $c_j$. 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. As such, the model after the first iteration of the RF algorithm is equivalent to a class-based bigram model which trains from the same corpus pre-segmented using frequency counts in the lexicon.

Testing is based on the Viterbi algorithm. The best segmentation and tagging $\mathcal{L}^*$ is retrieved from sentences of the test corpus, and the test-set perplexity is calculated from the log likelihood as follows.

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

where the summation is taken over all sentences $s_1^{T_i}$ in the corpus, and $L$ is the number of *characters* in the corpus. Each sentence boundary is counted as one character, since their transition are modeled. For simplicity, punctuations are used as sentence boundaries. Perplexity is used as a measure of the language model performance.

A corpus of daily newspaper articles is used as the training and testing set for the experiments. It is organized into sub-corpa of different sizes, as shown in Table 1. The JAN set is used for testing and others (FEB1 to FAUG) are used for training. The HMM and RF algorithms are applied to all the training sets, and parameters obtained after different iterations are used for testing. Their test set perplexities are shown in Figure 1. At iteration 0, i.e. based on the initial parameters, the test-set perplexity is 249.572.

As expected, the performance improves with the size of the training data. RF trained models are observed to converge quickly after a few iterations and remains stable, while HMM trained models attain a minimum in perplexity, but degrade on further iterations due to over-training. As the size of the training corpus increases, this minimum occurs at later iterations and with lower perplexity. Eventually it excels the RF model (Figure 2).

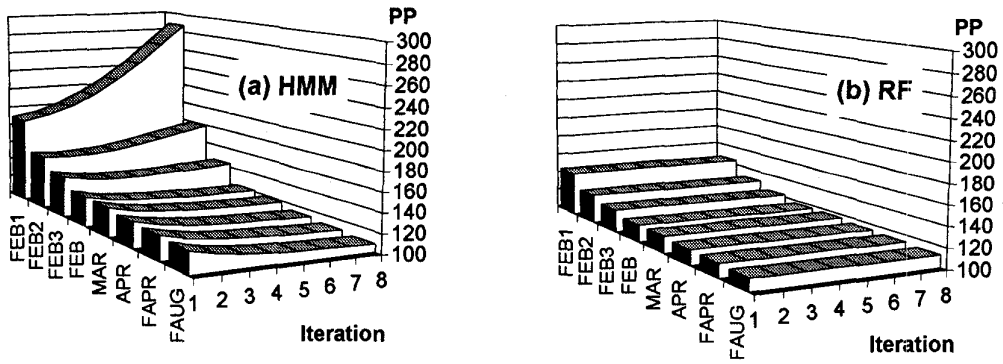


Figure 1: Test Set Perplexities of the JAN set trained by (a) HMM and (b) RF after different iterations

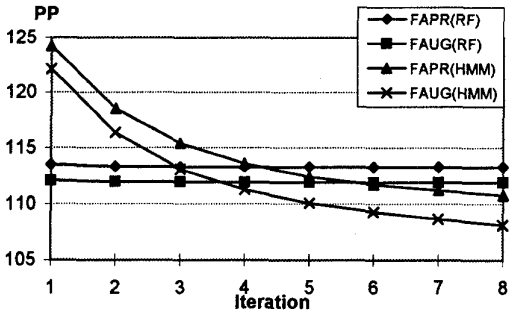


Figure 2: Comparison of HMM and RF trained models

| JAN | FEB1 | FEB2 | FEB3 | FEB | MAR | APR | FAPR | FAUG |
|---|---|---|---|---|---|---|---|---|
| 4227 | 98.5 | 250 | 504 | 1288 | 1962 | 3000 | 6250 | 21k |

Table 1: Sizes of Training/Testing Corpus ('000 Characters)

It is believed that the RF model performs better in case of insufficient data due to more robust smoothing. As the corpus size increases, the effect of smoothing is diminished, and obviously HMM training is superior as it is less 'greedy' and less easily got trapped at a local minimum as in RF training. That HMM out-performs RF only at relatively large training corpus (3m characters) may be due to the large number of parameters.

A further experiment is performed to use the models to decode phonetic inputs [6]. This is a difficult problem since each Chinese syllable can correspond to up to 80 different characters. Character recognition rates of 91.90% and 90.65% are obtained for models trained on the FAUG set using HMM and RF methods respectively.

## 6. CONCLUSION

In this paper a novel Ergodic Multigram HMM is introduced, whose application enables integrated, iterative training on untagged and unsegmented corpus for languages like Chinese. With enough training, iterative HMM training is shown to be superior to the $N$-gram method of the same order, estimated by iterative relative frequency counts. Further research can be done to investigate the effect of deleted-interpolation smoothing especially for limited training data, and extension to second order Markov modeling which gives a more accurate model. Other uses of the model, including phoneme to word conversion, can also be explored.

## 7. REFERENCES

[1] Brown, P.F. et al. (1992) "Class-Based $n$-gram Models of Natural Language". *Computational Linguistics*, 18:467-479.

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

[3] Merialdo, B. (1991), Tagging Text with a Probabilistic Model. *ICASSP 91*, pp.809-812.

[4] Schmid, H. (1994), Part-of-Speech Tagging with Neural Networks. *COLING 94*, pp.172-176.

[5] Chang, C.H., Chan, C.D. (1993) A Study on Integrating Chinese Word Segmentation and Part-of-Speech Tagging. *Comm. of COLIPS, Vol 3, No.1*, pp.69-77.

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

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

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

[9] Katz, S.M. (1987) Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustic Speech and Signal Processing, Vol 35*, pp.400-401.

[10] Chinese Knowledge Information Group (1993), *Technical Report No. 93-05*. Institute of Information Science, Academia Sinica, Taiwan.