Cross-Lingual Tense Tagging Based on Markov Tree Tagging Model

Yijiang Chen(✉), Tingting Zhu, Chang Su, and Xiaodong Shi

School of Information Science and Engineering, Xiamen University, Xiamen 361005, China
cyj@xmu.edu.cn

Abstract. In this paper, we transform the issue of Chinese-English tense conversion into the issue of tagging a Chinese tense tree. And then we propose Markov Tree Tagging Model to tag nodes of the untagged tense tree with English tenses. Experimental results show that the method is much better than linear-based CRF tagging for the issue.

Keywords: Chinese-English tense conversion · Markov tree tagging model · Tense tree · Untagged tense tree · Tagged tense tree

1 Related Work

Chinese-English machine translation systems need to deal with the issue of tense, otherwise it will affect the accuracy and fluency of translation results [1]. Gong etc. [2] used 2005 MT NIST data to find that tense translation has big effect.[1]

People have proposed several solutions to tense conversion. The common strategy is the use of manual rules [3] [4]. For example, if a temporal phrase is "昨天 (yesterday)", the sentence is tagged as "past" [4]. Ma [5] uses rules to acquire Chinese tenses and aspects, and then uses the predefined mapping relations to map them to English tenses. Methods based on manual rules are usually too rough and arbitrary.

Another strategy is the use of statistics methods [2] [5] [6].Yang Ye [5] uses the linear conditional random field (CRF) classifier to tag the verb sequence with English tenses. The advantage of this method is that it can cover complicated phenomenon of language, and consider tagging all verbs from the whole. Gong etc. [2] use N-grams of intra tense corpus and inter tense corpus to guess tenses. Gong etc. [6] use SVM method for tense classification. The method is combined with a statistical MT system.

In all these methods, interdependence of tenses between main clauses and subordinate clauses is not fully expressed. A compound sentence is composed of some clauses, and there are constraints of tenses between main clauses and subordinate clauses.

In this paper, we propose a method for Chinese-English tense conversion based on tense trees. We also propose a statistical machine learning method called Markov tree tagging model (MTTM). Firstly, we construct an untagged tense tree according to the parse tree of a Chinese sentence; secondly, we use MTTM to tag all nodes of the untagged tense tree with English tenses.

2 Tense Trees

The existing tense processing methods did not take full advantage of hierarchical information in parse trees. For this, we propose the concept of tense tree.

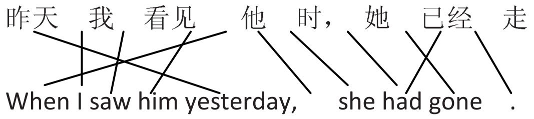
An untagged tense tree (UTT) is a reduced and flattened tree with syntactic and semantic information related to tense, which is converted from a parse tree.

A tagged tense tree (TTT) is an untagged tree plus tags, in which "IP, CP and verb" nodes are tagged with one of 16 English tenses and other nodes are tagged with "NONE". IP or CP is the label of a clause in a parse tree.

Untagged tense tree and tagged tree are collectively called tense tree. For example, Fig. 1 is bilingual word alignment between a Chinese sentence and its English translation. Fig. 2 is its untagged tense tree, and Fig. 3 is its tagged tense tree.

We believe that a clause (labeled with IP or CP) can have a tense, which is usually the tense of the main verb of the clause. We define the main verb of a clause (labeled with IP or CP) as the first verb which is directly covered by the label IP (or CP) of the clause. In Fig. 3, the verb "看见 (saw)" is the main verb of CP (but not IP).

Then we transform the issue of tense conversion into that of tagging tense tree.



昨天 我 看见 他 时， 她 已经 走
When I saw him yesterday, she had gone .

Fig. 1. Bilingual word alignment

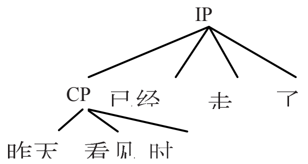


Fig. 2. Untagged tense tree

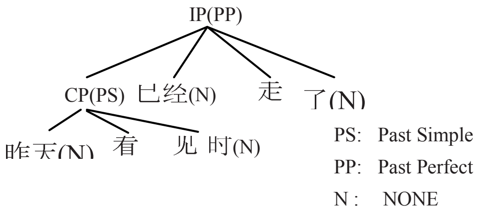


Fig. 3. Tagged tense tree

We construct the untagged tense tree during the traversal of the parse tree of a Chinese sentence. Attributes of nodes in a tense tree are as follows. (1) Attributes of a verb node include the verb itself, POS, whether being a modal verb, and its phase.(2) Attributes of a IP or CP node include the label(IP or CP), the label of the parent in the parse tree, and the type of the temporal phrase node in the clause. (3) Attributes of a temporal phrase node include the tense of the temporal phrase, point or period type of the temporal phrase, and whether occurring time earlier than reference time.(4) Attributes of a temporal adverb node include the adverb itself and its type [7]. (5)Attributes of a temporal auxiliary word node include the word itself, such as "了,着,过".

3 Markov Tree Tagging Model

3.1 Introduction to Tree Tagging Models

For the need to solve hierarchical issues, people try to tag nodes of a tree. For example, Fig. 4(a) is an untagged tree T, the tag $x_i \in H_i$. Fig. 4(b) is a tagged tree T[X].

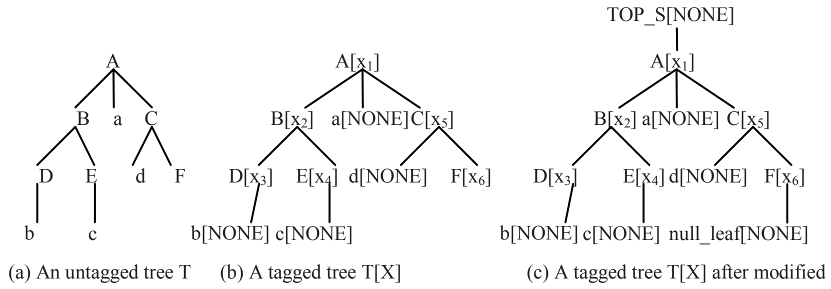


Fig. 4. A sample of tree tagging

Inspired by Head-Driven parsing method proposed by Collins [8], we propose a machine learning method called Markov tree tagging model (MTTM). The model has global consideration for best tagging. However, the algorithm of MTTM is very different from Head-Driven parsing method.

3.2 The Definition of MTTM

In order to make Markov tree tagging model (MTTM) more convenient to calculate the probability of a tree and to make the tree tagging algorithm more convenient to use recursion, we modify the untagged tree T as follows:(1) For a leaf node which we intend to tag (that is, the tag set is not {NONE}), we add a null_leaf node as its child. The null_leaf node has the only one tag NONE (that is, the tag set is {NONE}). (2) Add a root node TOP_S for the tree T. The tagging set of TOP_S is {NONE}. We can prove that the probability of the tree T is not changed. Shown as the Fig.4, we get Fig.4(c) after modifying Fig.4(b).

Tree T[X] can be generated by context free grammars. In the following statement, uppercase letters A, B, …, Z represent internal nodes (i.e. non-terminals) of the untagged tree, and lowercase letters a,b,…,w_1,w_2,…represent leaf nodes (i.e. terminals). In addition, we use the Greek alphabet $\alpha_1, \alpha_2, \ldots$ to represent any node (non-terminal or terminal). In Fig.4(b), a production of the non-terminal $\mathrm{A}[x_1]$ is $\mathrm{A}[x_1] \rightarrow \mathrm{B}[x_2]\mathrm{a}[\mathrm{NONE}]\mathrm{C}[x_5]$, a[NONE] is a terminal, and TOP_S[NONE] is a non-terminal.

MTTM is defined as follows: Assume there is an untagged tree T, in which the tag of the node i is $\mathrm{x}_i \in \mathrm{H}_i$, and tags of all internal nodes (not including leaf nodes because their tagging set is {NONE}) constitute the tagging vector X=$< \mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n > \in \mathrm{H}_1 \times \mathrm{H}_2 \times \ldots \times \mathrm{H}_n$. Then the best tag X^* of the tree T is

$$X^* = \underset{X\in H_1\times H_2\times\ldots\times H_n}{argmax} P(X/T) = \underset{X\in H_1\times H_2\times\ldots\times H_n}{argmax} P(T[X]/T)$$
$$= \underset{X\in H_1\times H_2\times\ldots\times H_n}{argmax} P(T[X]) = \underset{X\in H_1\times H_2\times\ldots\times H_n}{argmax} P(T[X]/TOP\_S[NONE])$$

In the following, we discuss how to decompose and calculate P(T[X]/ TOP_S[NONE]) and how to use dynamic programming algorithm to calculate $\underset{X}{max} P(T[X]/TOP\_S[NONE])$ and $\underset{X}{argmax} P(T[X]/TOP\_S[NONE])$ effectively.

3.3 Left-Most Derivation of the Tagged Tree T[X]

A tagged tree T[X] corresponds to a left-most derivation $r_1, r_2, \ldots, r_n$. We set the following rules: (1) The number i is the number of an internal node which corresponds to the left side of the production r_i. Node i is written as $Y_i[x_i]$. So we give each internal node a unique number. For example, TOP_S[NONE] is the node $Y_1[x_1]$. (2) A sub-tree of the tree T[X] whose root is $Y_i[x_i]$ is called $T_i[x_i]$. The leftmost derivation of the sub-tree $T_i[x_i]$ is $r_i, r_{i+1}, \ldots, r_j$.

Assuming $Y_i[x_i]$ is known, the sub-tree $T_i[x_i]$ is not unique because its internal node $Y_{i+1}[x_{i+1}]$, $Y_{i+2}[x_{i+2}]$, ⋯, $Y_j[x_j]$ will vary with different values of $x_{i+1}, \ldots, x_j$.

Obviously, a tagged tree T[X] = T_1[NONE], its root is $Y_1[x_1]$ = TOP_S [NONE], and its leftmost derivation is $r_1, r_2, \ldots, r_n$, that is, i =1, j=n. n is the number of internal nodes of T[X].

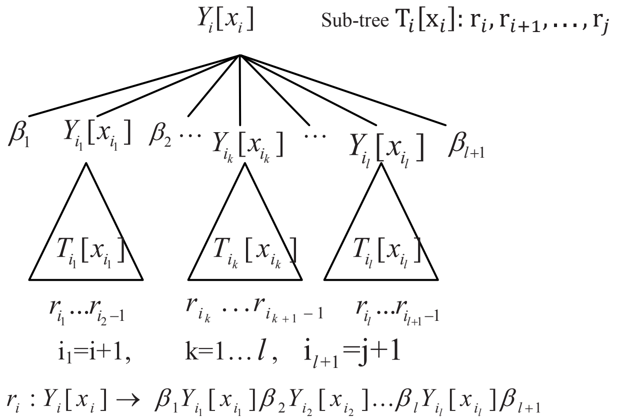

$Y_i[x_i]$
Sub-tree $T_i[x_i]$: $r_i, r_{i+1}, \ldots, r_j$
β_1 $Y_{i_1}[x_{i_1}]$ $\beta_2 \cdots$ $Y_{i_k}[x_{i_k}]$ $\cdots$ $Y_{i_l}[x_{i_l}]$ β_{l+1}
$T_{i_1}[x_{i_1}]$
$T_{i_k}[x_{i_k}]$
$T_{i_l}[x_{i_l}]$
$r_{i_1} \ldots r_{i_2 -1}$
$r_{i_k} \ldots r_{i_{k+1}-1}$
$r_{i_l} \ldots r_{i_{l+1}-1}$
i_1=i+1, k=1…l, i_{l+1}=j+1
$r_i : Y_i[x_i] \rightarrow \beta_1 Y_{i_1}[x_{i_1}]\beta_2 Y_{i_2}[x_{i_2}]\ldots\beta_l Y_{i_l}[x_{i_l}]\beta_{l+1}$

Fig. 5. Leftmost derivation of $T_i[x_i]$

Assume the production r_i: $Y_i[x_i] \rightarrow \beta_1 Y_{i_1}[x_{i_1}]\beta_2 Y_{i_2}[x_{i_2}] \ldots \beta_l Y_{i_l}[x_{i_l}]\beta_{l+1}$, β_q is a string which is composed of one or more terminals, q=1…l +1, (β_q can be null), $Y_{i_k}[x_{i_k}]$ is the kth non-terminal of the right hand side of the production r_i, k=1... l, and the number of its corresponding internal node in tree T[X] is i_k. l is the number of the non-terminals of the right hand side of the production r_i. When l =0, production r_i has no non-terminal. Because the production sequence $\mathrm{r}_i, \mathrm{r}_{i+1}, \ldots, \mathrm{r}_j$ is the left-most derivation of the tree $\mathrm{T}_i[\mathrm{x}_i]$ which begins with the non-terminal $\mathrm{Y}_i[\mathrm{x}_i]$, r_{i+1} is the direct derivation of the non-terminal $\mathrm{Y}_{i_1}[\mathrm{x}_{i_1}]$, that is r_{i+1}: $\mathrm{Y}_{i_1}[\mathrm{x}_{i_1}] \to \lambda$, and i_1=i+1. Let $i_{l+1} = j+1$, then $r_{i_k} \ldots r_{i_k-1}$is left-most derivation of the sub-tree $\mathrm{T}_{i_k}[\mathrm{x}_{i_k}]$ which begins with the non-terminal $\mathrm{Y}_{i_k}[\mathrm{x}_{i_k}]$. Details are shown as Fig.5.

3.4 Using Dynamic Programming to Find the Best Tagged Tree

In the following, we discuss how to use dynamic programming to find the best tagged tree. We use dynamic programming twice, represent with σ and δ respectively, and use the method of accumulator.

$\sigma(\mathrm{Y}_i[\mathrm{x}_i])$ = the maximum probability of the sub-tree $\mathrm{T}_i[\mathrm{x}_i]$ when $\mathrm{x}_i \in \mathrm{H}_i$ is known That is, $\sigma(\mathrm{Y}_i[\mathrm{x}_i]) = {}_{x_{i+1}\in H_{i+1}, x_j \in H_j}^{max} P(\mathrm{T}_i[\mathrm{x}_i]/\mathrm{Y}_i[\mathrm{x}_i]\ )$

Assume $\hat{T}_i[\mathrm{x}_i]$ is the sub-tree which maximizes the probability $P(\mathrm{T}_i[\mathrm{x}_i]/\mathrm{Y}_i[\mathrm{x}_i])$. That is, $\hat{T}_i[\mathrm{x}_i] = {}_{x_{i+1}\in H_{i+1}, x_j \in H_j}^{argmax} P(\mathrm{T}_i[\mathrm{x}_i]/\mathrm{Y}_i[\mathrm{x}_i])$.

Specially, when $\mathrm{i}=1, \mathrm{Y}_1 = TOP\_S, x_1 \in H_1 = \{NONE\}, T[X] = T_1[NONE]$, then probability of the best tagged tree T[X*] is

$$\begin{aligned} &{}_{X\in H_1 \times H_2 \times \ldots \times H_n}^{max} P(T[X]/\mathrm{TOP\_S[NONE]}) \\ &= {}_{x_2\in H_2, \ldots, x_m \in H_n}^{max} P(\mathrm{T}_1[\mathrm{NONE}]/\ \mathrm{TOP\_S[NONE]}) = \sigma(\mathrm{Y}_1[\mathrm{x}_1]) \end{aligned} \tag{1}$$

And, $$T[X^*] = {}_{X\in H_1 \times H_2 \times \ldots \times H_n}^{argmax} P(T[X]/TOP\_S[NONE])$$

$$= {}_{x_2\in H_2, \ldots, x_n \in H_n}^{argmax} P(\mathrm{T}_1[\mathrm{NONE}]/\ \mathrm{TOP\_S[NONE]}) = \hat{T}_1[\mathrm{x}_1] \tag{2}$$

That is to say, $\hat{T}_1[\mathrm{x}_1]$ is the best tagging tree, and its probability is $\sigma(\mathrm{Y}_1[\mathrm{x}_1])$. We use a recursive method to post-order traverse the untagged tree T, and calculate $\sigma(\mathrm{Y}_1[\mathrm{x}_1])$ and $\hat{T}_1[\mathrm{x}_1]$, $\sigma(\mathrm{Y}_2[\mathrm{x}_2])$and $\hat{T}_2[\mathrm{x}_2]$,..., $\sigma(\mathrm{Y}_n[\mathrm{x}_n])$ and $\hat{T}_n[\mathrm{x}_n]$. Note that, $\sigma(\mathrm{Y}_1[\mathrm{x}_1])$ and $\hat{T}_1[\mathrm{x}_1]$ are the last ones to calculate, because $\mathrm{Y}_1[\mathrm{x}_1] = \mathrm{TOP\_S[NONE]}$ is the root of tree T, which is the last node to visit during the post-order traversal.

In the following, we discuss how to calculate $\sigma(\mathrm{Y}_i[\mathrm{x}_i])$ and $\hat{T}_i[\mathrm{x}_i]$. Because
$\mathrm{P}(\mathrm{T}_i[\mathrm{x}_i]/\mathrm{Y}_i[\mathrm{x}_i]) = \mathrm{P}(r_i, r_{i+1} \ldots, r_j/Y_i[x_i]) = \mathrm{P}(r_i/Y_i[x_i]) * \mathrm{P}(r_{i+1} \ldots, r_j/r_i, Y_i[x_i])$
$\approx \mathrm{P}(r_i/Y_i[x_i]) * \prod_{k=1}^{l} P(r_{i_k} \ldots r_{i_k-1}/r_i, Y_i[x_i])$ // ≈ is independence assumption

$$\approx \mathrm{P}(r_i/Y_i[x_i]) * \prod_{k=1}^{l} P(r_{i_k}, \ldots, r_{i_k+1}/Y_{i_k}[x_{i_k}])$$

$$= P(r_i/Y_i[x_i]) * \prod_{k=1}^{l} P(T_{i_k}[x_{i_k}]/Y_{i_k}[x_{i_k}])$$

Then $\sigma(Y_i[x_i]) = {}_{x_{i+1}\in H_{i+1}, x_j \in H_j}^{max} P(\mathrm{T}_i[\mathrm{x}_i]/\mathrm{Y}_i[\mathrm{x}_i])$

$$\approx \max_{x_{i+1}\in H_{i+1}, x_j \in H_j} P(r_i/Y_i[x_i]) * \prod_{k=1}^{l} P\left(T_{i_k}[x_{i_k}]/Y_{i_k}[x_{i_k}]\right)$$

$$= \max_{x_{i_k}\in H_{i_k}, k=1\ldots l}\{P(r_i/Y_i[x_i]) * \prod_{k=1}^{l} \max_{x_{i_k+1}\in H_{i_k+1},\ldots,x_{i_{k+1}-1}\in H_{i_{k+1}-1}} \{P\left(T_{i_k}[x_{i_k}]/Y_{i_k}[x_{i_k}]\right)\}$$

$$= \max_{x_{i_k}\in H_{i_k}, k=1\ldots l}\{P(r_i/Y_i[x_i]) * \prod_{k=1}^{l} \sigma\left(Y_{i_k}[x_{i_k}]\right)\} \quad (3)$$

We decompose r_i using Markov chain to calculate $P(r_i/Y_i[x_i])$ in (3) as follows.

Assume r_i is: $Y_i[x_i] \to \alpha_1^i[x_1^i]\alpha_2^i[x_2^i]\ldots\alpha_m^i[x_m^i]$, $\alpha_q^i[x_q^i]$ can be either the internal node (i.e., non-terminal) or the leaf node (i.e. terminal), q=1,2,…,m. m is the number of non-terminals in the right hand side of r_i . We add two terminals STOP in the right hand side of the rule. That is,

$$\alpha_0^i[x_0^i] = STOP[NONE], \alpha_{m+1}^i[x_{m+1}^i] = STOP[[NONE],$$
$$r_i: Y_i[x_i] \to \alpha_0^i[x_0^i]\alpha_1^i[x_1^i]\alpha_2^i[x_2^i]\ldots\alpha_m^i[x_m^i]\alpha_{m+1}^i[x_{m+1}^i].$$

This modification will not change the probability of the production and the tree.

Then $\sigma(Y_i[x_i]) = \max_{x_{i_k}\epsilon H_{i_k}, k=1\ldots l}\{P(r_i/Y_i[x_i]) * \prod_{k=1}^{l}\sigma\left(Y_{i_k}[x_{i_k}]\right)\}$

$$= \max_{x_{i_k}\epsilon H_{i_k}, k=1\ldots l}\{P(\alpha_0^i[x_0^i]\alpha_1^i[x_1^i]\alpha_2^i[x_2^i]\ldots\alpha_m^i[x_m^i]\alpha_{m+1}^i[x_{m+1}^i]/Y_i[x_i]) \prod_{k=1}^{l}\sigma\left(Y_{i_k}[x_{i_k}]\right)\}$$

$$\approx \max_{x_{i_k}\epsilon H_{i_k}, k=1\ldots l}\{\{P(\alpha_0^i[x_0^i]/Y_i[x_i]) \prod_{q=1}^{m+1} P\left(\alpha_q^i[x_q^i]/Y_i[x_i], \alpha_{q-1}^i[x_{q-1}^i]\right)\} \prod_{k=1}^{l}\sigma\left(Y_{i_k}[x_{i_k}]\right)\}$$

$$= \max_{x_{i_k}\epsilon H_{i_k}, k=1\ldots l}\{\{\textstyle\prod_{q=1}^{m+1} P\left(\alpha_q^i[x_q^i]/Y_i[x_i], \alpha_{q-1}^i[x_{q-1}^i]\right)\} * \prod_{k=1}^{l}\sigma\left(Y_{i_k}[x_{i_k}]\right)\} \quad (4)$$

Let $\rho\left(\alpha_q^i[x_q^i]\right) = \begin{cases} 1 & when\ \alpha_q^i[x_q^i]\ is\ a\ leaf\ node \\ \sigma\left(Y_{i_k}[x_{i_k}]\right) & when\ \alpha_q^i[x_q^i] = Y_{i_k}[x_{i_k}] \end{cases} \quad q = 1,2,\ldots,m+1$

Then formula (4) is as follows

$$\sigma(Y_i[x_i]) = \max_{x_{i_k}\epsilon H_{i_k}, k=1\ldots l} \prod_{q=1}^{m+1}\{P\left(\alpha_q^i[x_q^i]/Y_i[x_i], \alpha_{q-1}^i[x_{q-1}^i]\right)\rho\left(\alpha_q^i[x_q^i]\right)\}$$

$$= \max_{x_q^i\epsilon H_q^i, q=1,2,\ldots,m+1} \prod_{q=1}^{m+1}\{P\left(\alpha_q^i[x_q^i]/Y_i[x_i], \alpha_{q-1}^i[x_{q-1}^i]\right)\rho\left(\alpha_q^i[x_q^i]\right)\} \quad (5)$$

In the following, we use dynamic programming again to calculate the value of formula (5). This step is similar to the Viterbi algorithm of HMM, because it is to get the best tagging $< x_1^i, x_2^i, \ldots x_m^i, x_{m+1}^i >$ of the sequence $< \alpha_1^i, \alpha_2^i, \ldots \alpha_m^i, \alpha_{m+1}^i >$.

(1) Initialization

$$\delta_0\,(Y_i[x_i], x_0^i) = 1, x_0^i \epsilon \{NONE\}$$

(2) Derivation

For $q = 1,2,\ldots,m+1, x_q^i \epsilon H_q^i$,

$$\delta_{\mathrm{i}}\left(Y_i[x_i], x_q^i\right) = \max_{x_{q-1}^i \epsilon H_{q-1}^i} \delta_{q-1}\left(Y_i[x_i], x_{q-1}^i\right) * P\left(\alpha_q^i\left[x_q^i\right] / Y_i[x_i], \alpha_{q-1}^i\left[x_{q-1}^i\right]\right) * \rho\left(\alpha_q^i\left[x_q^i\right]\right)$$

Storage the backtracking path

$$\psi_q\left(Y_i[x_i], x_q^i\right) = \operatorname{argmax}_{x_{q-1}^i \epsilon H_{q-1}^i} \delta_{q-1}\left(Y_i[x_i], x_{q-1}^i\right) * P\left(\alpha_q^i\left[x_q^i\right] / Y_i[x_i], \alpha_{q-1}^i\left[x_{q-1}^i\right]\right) * \rho\left(\alpha_q^i\left[x_q^i\right]\right)$$

(3) Assume the most likely sequence of the backtracking is

$\hat{x}_0^i, \hat{x}_1^i, \ldots, \hat{x}_m^i, \hat{x}_{m+1}^i$, then,

$$\hat{x}_{m+1}^i = \operatorname{argmax}_{x_{m+1}^i \epsilon H_{m+1}^i} \delta_{m+1}\left(Y_i[x_i], x_{m+1}^i\right) \text{ (Actually, } x_{m+1}^i \epsilon H_{m+1}^i = \{NONE\})$$

$$\hat{x}_{q-1}^i = \psi_q\left(Y_i[x_i], x_q^i\right), q = m+1, m, \ldots, 1$$

Now, the maximum probability of the sub-tree $T_i[x_i]$ whose root is $Y_i[x_i]$ is

$$\sigma(Y_i[x_i]) = \max_{x_{m+1}^i \epsilon H_{m+1}^i} \delta_{m+1}\left(Y_i[x_i], x_{m+1}^i\right)$$

Then $\sigma(Y_i[x_i]) = \delta_{m+1}\left(Y_i[x_i], x_{m+1}^i = NONE\right)$ because $x_{m+1}^i \epsilon H_{m+1}^i = \{NONE\}$

So, under the circumstance that the value of x_i is known, $\psi(Y_i[x_i]) =< \hat{x}_0^i, \hat{x}_1^i, \ldots, \hat{x}_m^i, \hat{x}_{m+1}^i >$ is the best tag of r_i in the sub-tree $\hat{T}_i[x_i]$whose root is $Y_i[x_i]$.

Assume that the number of internal nodes of the untagged tree is N, the maximum size of the tagging sets is H=max(|H_i|), and the maximum number of child nodes of an internal node (including STOP node) is M, then the time complexity is O(N*M*H^3).

4 Use Markov Tree Tagging Model to Tag Tense Trees

Training:

(1) Construct the untagged tense tree according to the parse tree of a sentence.

(2) Manually tag tenses of IP, CP and verb nodes of the untagged tense tree T, and construct the tagged tense tree T[X].

(3) Calculate probability parameters, which will be used to calculate the probability of a tagged tense tree in decoding.

Decoding:

(1) Construct the untagged tense tree according to the grammar tree of a sentence.
(2) Use MTTM to get the tense tree T[X*] as the final result.

$$X^* = \underset{\substack{\text{the English tense of IP,CP,verb node} \\ \in \text{t aggi ng set}}}{\operatorname{argmax}} P(T[X]/T) = \underset{\substack{\text{he English tense of IP,CP,verb node} \\ \in \text{t aggi ng set}}}{\operatorname{argmax}} P(T[X])$$

The tagging set of IP, CP and verb nodes is {16 tenses}, and that of other nodes is {NONE}. Attribute values of node i in T can be regarded as Y_i in MTTM.

5 Experimental Results and Discussion

5.1 Experimental Data

We manually tag *Chinese TreeBank 6.0* with 16 English tenses according to its English translation *English Chinese Translation Treebank v 1.0*. The chtb_0110.fid ~ chtb_0139.fid is training set, and chtb_0140.fid ~ chtb_0144.fid is the test set.

We use accuracy [5] to measure the whole performance, and use precision, recall and F_measure to measure the performance of each tense.

5.2 Experimental Results

We use MTTM to tag untagged tense trees of the test set with 16 English tenses, convert the experimental results into the 3 tenses (past, present, future), and compare them with the experimental results of Ye [5], because they only contain 3 tenses.

Table 1 is the comparison of the verb tagging results between our method and the method of Ye [5]. The accuracy of our method is better than that of Ye. The precision and recall of present tense are very low in Ye [5], and our method has a substantial increase. Because future tense appears rarely in the tense corpus and the test set, the F-measure of future tense of Yang Ye and our method are very low.

Table 1. The experimental comparison of tagging verbs between our method and Yang Ye [5]

	Accuracy of Ye[5]=58.21%			Accuracy of our method = 66.0%		
	Precision	Recall	F-measure	Precision	Recall	F-measure
past	67.57%	79.55%	72.10%	55.1%	81.4%	65.7%
present	42.50%	27.48%	32.07%	80.5%	61.9%	70.01%
future	29.66%	25.56%	21.56%	38.5%	22.7%	28.6%

6 Conclusions

The experimental results show that the verb accuracy of tense tagging based on tense tree improves 7.79% compared to that of Ye which is based on sequence. This comparison shows that hierarchical interdependence is important to tense tagging, while methods based on sequence are difficult to express hierarchical interdependence. Though linear conditional random field is a very good machine learning method, the correct expression of hierarchical interdependence is more important.

Acknowledgments. Supported by the National Natural Science Foundation of China (No. 61075058).

References

1. Liu, Q., Yu, S.: Discussion on the difficulties of Chinese-English machine translation. In: Huang, C.L. (ed.) International Conference on Chinese Information Processing, pp. 507–514. Tsinghua University Press, Beijing (1998)
2. Gong, Z., Zhang, M., Tan, C., Zhou, G.: N-gram-based tense models for statistical machine translation. In: Proceedings of EMNLP, East Stroudsburg, PA, USA, pp. 276–285. Association for Computational Linguistics (ACL) (2012)
3. Chen, J., Dai, X., Chen, J., Wang, Q.: Processing of Tense and Aspect in Chinese-English Machine Translation. Application Research of Computers **21**(3) (2004)
4. Hongmei, M.: Research on the Representation and Application of Chinese Context in Chinese-English Machine Translation [Ph.D. Thesis], Changsha, Hunan: National University of Defense Technology (2002)
5. Ye, Y., Zhang, Z.: Tense tagging for verbs in cross-lingual context: a case study. In: Dale, R., Wong, K.-F., Su, J., Kwong, O.Y. (eds.) IJCNLP 2005. LNCS (LNAI), vol. 3651, pp. 885–895. Springer, Heidelberg (2005)
6. Gong, Z., Zhang, M., Tan, C., Zhou, G.: Classifier-based tense model for SMT. In: Proceedings of the 24th International Conference on Computational Linguistics, PP. 411–420 (2012)
7. Lu, J., Ma, Z.: Comments on Function Word of Modern Chinese, pp. 106–141. Peking University Press, Beijing (1999)
8. Collins, M.: Head-Driven Statistical Models for Natural Language Parsing. Computational Linguistics (2003)