Firstly, we collect chunk type transition information between chunk types by observing every pair of adjacent chunks in the training corpus, and record a chunk type transition matrix. For example, from the Chinese Treebank that we used for our experiments, a transition from chunk type ADJP to ADVP does not occur in the training corpus, the corresponding matrix element is set to *false*, *true* otherwise. During decoding, the chunk type transition information is used to prune unlikely combinations between current chunk and the preceding chunk by their chunk types.

Secondly, a POS tag dictionary is used to record POS tags associated with each chunk type. Specifically, for each chunk type, we record all POS tags appearing in this type of chunk in the training corpus. During decoding, a segment of continuous words that contains only allowed POS tags according to the POS tag dictionary will be considered to be a valid chunk candidate.

Finally, the system records the maximum number of words for each type of chunk in the training corpus. For example, in the Chinese Treebank, most types of chunks have one to three words. The few chunk types that are seen with length bigger than ten are NP, QP and ADJP. During decoding, the chunk candidate whose length is greater than the maximum chunk length associated with its chunk type will be discarded.

For the above pruning schemes, development tests show that it improves the speed significantly, while having a very small negative influence on the accuracy.

5 Learning

5.1 Discriminative Online Training

By defining features, a candidate output y is mapped into a global feature vector, in which each dimension represents the count of a particular feature in the sentence. The learning task is to set the parameter values w using the training examples as evidence.

Online learning is an attractive method for the joint model since it quickly converges within a few iterations (McDonald, 2006). We focus on an online learning algorithm called MIRA, which is a relaxed, online maximum margin training algorithm with the desired accuracy and scalability properties (Crammer, 2004). Furthermore, MIRA is very flexible with respect to the loss function. Any loss function on the output is compatible with MIRA since it does not require the loss to factor according to the output, which enables our model to be optimized with respect to evaluation metrics directly. Figure 2 outlines the generic online learning algorithm (McDonald, 2006) used in our framework.

MIRA updates the parameter vector w with two constraints: (1) the positive example must have a higher score by a given margin, and (2) the change to w should be minimal. This second constraint is to reduce fluctuations in w. In particular, we use a generalized version of MIRA (Crammer et al., 2005; McDonald, 2006) that can incorporate *k-best* decoding in the update procedure.

Input: Training set $S = \{(x_t, y_t)\}_{t=1}^{T}$
1: $w^{(0)} = 0$; $v = 0$; $i = 0$
2: for *iter* = 1 to N do
3: for $t = 1$ to T do
4: $w^{(i+1)}$ = update $w^{(i)}$ according to (x_t, y_t)
5: $v = v + w^{(i+1)}$
6: $i = i + 1$
7: end for
8: end for
9: $w = v/(N \times T)$
Output: weight vector w

Figure 2: Generic Online Learning Algorithm

In each iteration, MIRA updates the weight vector w by keeping the norm of the change in the weight vector as small as possible. Within this framework, we can formulate the optimization problem as follows (McDonald, 2006):

$$w^{(i+1)} = \arg\min_{w} \left\| w - w^{(i)} \right\|$$
$$s.t. \ \forall y' \in best_k(x_t; w^{(i)}): \quad (2)$$
$$w^T \cdot \Phi(y_t) - w^T \cdot \Phi(y') \geq L(y_t, y')$$

where $best_k(x_t; w^{(i)})$ represents a set of top *k-best* outputs for x_t given the weight vector $w^{(i)}$. In our implementation, the top *k-best* outputs are obtained with a straightforward *k-best* extension to the decoding algorithm in section 4.1. The above quadratic programming (QP) problem can be solved using Hildreth's algorithm (Yair Censor, 1997). Replacing Eq. (2) into line 4 of the algorithm in Figure 2, we obtain *k-best* MIRA.

As shown in (McDonald, 2006), parameter averaging can effectively avoid overfitting. The