Easy-First Chinese POS Tagging and Dependency Parsing

Ji Ma, Tong Xiao, Jingbo Zhu, Feiliang Ren
Natural Language Processing Laboratory
Northeastern University, China

majineu@outlook.com, zhujingbo@mail.neu.edu.cn,
xiaotong@mail.neu.edu.cn, renfeiliang@ics.neu.edu.cn

ABSTRACT

The easy-first non-directional dependency parser has demonstrated its advantage over transition based dependency parsers which parse sentences from left to right. This work investigates easy-first method on Chinese POS tagging, dependency parsing and joint tagging and dependency parsing. In particular, we generalize the easy-first dependency parsing algorithm to a general framework and apply this framework to Chinese POS tagging and dependency parsing. We then propose the first joint tagging and dependency parsing algorithm under the easy-first framework. We train the joint model with both supervised objective and additional loss which only relates to one of the individual tasks (either tagging or parsing). In this way, we can bias the joint model towards the preferred task. Experimental results show that both the tagger and the parser achieve state-of-the-art accuracy and runs fast. And our joint model achieves tagging accuracy of 94.27 which is the best result reported so far.

KEYWORDS: dependency parsing, POS tagging, perceptron, easy-first

Proceedings of COLING 2012: Technical Papers, pages 1731–1746,
COLING 2012, Mumbai, December 2012.

1 Introduction

To sequential labelling problems, such as POS tagging or incremental parsing, traditional approaches 1) decompose the input sequence into several individual items each of which corresponds to a token of the input sequence; 2) predict these items in a fixed left-to-right (or right-to-left) order (Collins 2002; Ratnaparkhi 1996). The drawback of such fixed order approach is that when predicting one item, only the labels on the left side can be used while the labels on the right side is still unavailable. (Goldberg and Elhadad, 2010) proposed the easy-first dependency parsing algorithm to relax the fixed left-to-right order and to incorporate more structural features from both sides of the attachment point. Comparing with a deterministic transition based parser which parses the sentence left-to-right, their deterministic easy-first parser achieves significant better accuracy.

The key idea behind their parsing algorithm is to always process the easy attachments in the early stages. The hard ones are delayed to late stages until more structural features on both sides of the attachment point are accessible so as to make more informed decision. In this work, we further generalize the algorithm of (Goldberg and Elhadad, 2010) to a general sequential labelling framework. We apply this framework to Chinese POS tagging and dependency parsing which has never been studied under the easy-first framework before.

One characteristic of Chinese dependency parsing is that parsing performance can be dramatically affected by the quality of POS tags of the input sentence (Li et al., 2010). Recent work (Li et al., 2010; Hatori et al., 2011; Bohnet and Nivre, 2012) empirically verified that solving POS tagging and dependency parsing jointly can boost the performance of both the two tasks. To further improve tagging and parsing accuracy, we also solve the two tasks jointly. While previous joint methods are either graph-based or transition-based algorithm, in this work we propose the first joint tagging and dependency parsing algorithm under the easy-first framework. In addition, we also adopt a different training strategy to learn the model parameters. Previous approaches all train their joint model with the objective of minimizing the loss between the reference and the predicted output. Those methods make no distinction between the loss caused by POS tagging and the loss caused by dependency parsing. Though such objective is proper for minimizing the total loss, it may not be optimal in terms of pursuing the best result of a certain individual task (neither tagging nor dependency parsing). To this end, our training method also incorporates additional loss which relates to only one of the individual tasks. And the losses are iteratively optimized on the training set. In this way we can bias the joint model towards the preferred task. Similar techniques have been used in parser domain adaptation (Hall et al., 2011). However, to our knowledge, no one has done this in joint tagging and dependency parsing before.

Experimental results show that under the easy-first framework, even deterministic tagger and parser achieve quite promising performance. For Chinese POS tagging, the tagger achieves an accuracy of 93.84[1] which is among the top Chinese taggers reported so far. Moreover, the tagging speed is about 2000 sentences per-second, much faster than the state-of-the-art taggers (Li et al., 2010; Hatori et al., 2011). For Chinese dependency parsing, when the input sentence is equipped with automatically assigned POS tags, the parser achieves an unlabelled score of 77.66 and runs at the speed of more 300 sentences per second. Such accuracy is among the state-of-the-art transition-based dependency parsers even those parsers are enhanced with beam

1 On the same data set, the best tagger so far reported achieves an accuracy of 93.82. Joint methods yield higher accuracy, but are not directly comparable.

search (section 4). For joint tagging and dependency parsing, we achieve significant improvement on both the two sub-tasks. In particular, we achieve the tagging accuracy of 94.27 which is the highest score reported on the same data set so far.

2 Easy-First POS tagging, dependency parsing and joint tagging and parsing

In this section, we first describe a generalized easy-first sequential labelling framework. Then we show how to apply this framework to the task of POS tagging and dependency parsing. Finally, we propose the joint POS tagging and dependency parsing algorithm under this framework.

2.1 Generalized easy-first sequential labelling algorithm

For solving sequential labelling problems, the first step is to decompose the input sequence into a list of small items $t_1, t_2, \ldots, t_n$. The items are then labelled separately. This list of items is the input of the generalized algorithm. In addition, the algorithm also requires a set of task specific labels $\{l_1, l_2, \ldots, l_m\}$ and a labelling function. Take POS tagging for example, t_i corresponds to the i-th word of the input sentence and the label set corresponds to the set of POS tags. The labelling function associates a word with a certain POS tag.

The pseudo-code of the algorithm is shown in Algorithm 1. Initially, all the items are marked as unprocessed (line 2). Then, the algorithm solves each of the unprocessed item t by labelling t with a suitable label l. After that, t is marked as processed. This process repeats until no items left unprocessed (line 3 to line 8).

One thing to note is that the small items are not processed in a $t_1, t_2 \ldots$ to t_n order. Instead, at each step the algorithm automatically chooses an item-label pair according to a scoring function $score(t, l)$ (line 4). This function is not only responsible for selecting a correct label for a certain item but also responsible for determining the order in which each of the items are processed. Ideally, the scoring function prefers to process easy items in the early stages and delay the hard ones to late stages. When dealing with the hard ones, the label information built on both sides of the current item become accessible. In this way, more informed prediction can be made and the extent of error propagation can be limited (Goldberg and Elhadad, 2010).

Algorithm 1 is a generalization of the easy-first parsing algorithm of (Goldberg and Elhadad, 2010). This generalized version can be naturally instantiated to a wide verity of applications.

2.2 Easy-first POS tagging

In this section, we show how to instantiate algorithm 1 into a POS tagger. The problem of POS tagging is to find a sequence of POS tags for the input sentence. For this problem,

- t_i corresponds to i-th word, w_i, of the input sentence.
- L corresponds to the POS tag set, **POS**. We use x to denote a certain tag in **POS.**
- The labelling function, $\text{labelling}^{pos}(w_i, x)$, set x as the POS tag of w_i

A concrete example of tagging the sentence "中国/China 对/to 外/outside world 开放/open 稳步/steadily 前行/advance" ("China's opening up steadily advances") is shown in figure 1. The challenging part of this sentence is w_4, "开放/opening up", which can be either a VV (verb) or a NN (noun). If we process this sentence in a left-to-right order, the word "开放" would be quite difficult. In the easy-first framework, this can be easily handled with the following steps.

Algorithm 1: Generalized Easy-First Sequential Labelling	
Input	T= t_1, t_2, …t_n: a sequence of items to be labelled L={l_1, l_2, …, l_m}: a set of task specific labels labelling: task specific labelling function
1	$nProcessed \leftarrow 0$
2	set each t_i as unprocessed
3	**repeat**
4	$(\bar{t}, \bar{l}) \leftarrow \text{argmax}_{l \in L, t.unprocessed=true} score(t, l)$
5	labelling $(\bar{t}, \bar{l})$
6	set $\bar{t}$ as processed
7	$nProcessed \leftarrow nProcessed + 1$
8	**until** nProcessed = |T|

Initially (step 0), all the six words w_1, …,w_6 are marked as unprocessed. Each step, the tagger enumerates all possible (w, x) pairs and chooses the most confident one according to the scoring function. Since the word "中国/China" always occurs as a NR (proper noun) in the training set, thus at the step 1, ("中国", NR) is selected[2] and the tagger tags "中国" with NR. After this step, "中国" is marked as processed.

At step 2, for those **unprocessed** words, the tagger re-computes the scores of all (w, x) pairs based on the local context, surrounding words and tags within a window, and selects the one with the highest score to deal with. This time, ("外", NN) is selected and the tagger tags "外" as a NN. Similarly, at step 3, the tagger assigns tag P (preposition) to word "对/to". Step 4 and so on.

At the last step (step 6), the only unprocessed word is w_4, "开放". Since for Chinese, an adverb always precede the verb which it modifies and succeeds the noun which is the subject. Therefore, based on the following tags o_5:AD and o_6:VV, the tagger can easily infer that the correct tag for "开放" is NN. After "开放" is tagged, all words are processed and the tagging procedure stops.

We see algorithm 1 can be easily instantiated to POS tagging, a relatively simple task. In the next sections, we show how to apply algorithm 1 to more complicated tasks.

2.3 Easy-first dependency parsing

The easy-first dependency parsing algorithm was originally proposed by (Goldberg and Elhadad, 2010), we re-describe it here for completeness and also to illustrate how algorithm 1 can be instantiated to dependency parsing.

Given an input sentence of n words, the task of dependency parsing is: for each word, find its lexical head which it modifies. One exception is the **head word** of the sentence which does not modify any other word. All the others each modify exactly one word and no one modifies itself. For dependency parsing:

- t_i corresponds to i-th word, w_i, of the input sentence.
- L corresponds to an action set **ACT** which contains three actions {attach_Left, attach_Right, set_Root}.

Note for dependency parsing, each label corresponds to an action which is designed to manipulate a list of partial analysis p_1,…, p_n, called pending (Goldberg and Elhadad, 2010). p_i re-

[2] At each step, the selected item is boldfaced. The underlined items are those marked as processed.

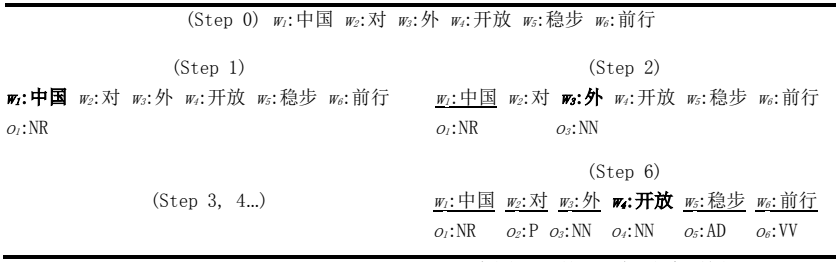

FIGURE 1 – A trace of easy-first tagger for sentence "中国 对 外 开放 稳步 前行" ("China's opening up steadily advances"). o_i denotes the POS tag of w_i

-cords the dependency tree rooted at w_i and p_i is initialized with w_i.

The main loop of easy-first dependency parsing is shown in figure 2. Each step, the parser computes the scores of all possible (p, a) pairs based on local context, surrounding partial analysis within a window, and then selects the best pair to feed to the labelling function, labelling^{dep}. Then labelling^{dep} performs the given action. In particular, labelling^{dep} (p_i, attach_Left) set p_i as the child of its left neighbour in the pending list. labelling^{dep} (p_i, attach_right) set p_i as the child of its right neighbour in the pending list. After that, the selected partial analysis p is marked as processed and removed from the pending list (line 7 to line 8).

Since at each step, one partial analysis is removed from the pending list, after $n-1$ steps, the pending list only contains one item, say p_h, which is the dependency tree of the whole sentence. In such satiation, (p_h, set_Root) is enforced to be the only choice and labelling^{dep} (p_h, set_Root) sets the root of p_h which is w_h as the head of the sentence.

An example of parsing the sentence "中国 对 外 开放 稳步 前行" is shown in figure 3. At the first step, (p_3, attach_Left) is selected and fed to the labelling^{dep} function. The labelling function set p_3 as the child of its left neighbour p_2 as shown in figure 2 (1). p_3 is then removed from the pending list. Note that, after p_3 is removed, p_2 and p_4 become neighbours. The following steps are executed in a similar way except for step 6 where only p_6 is left in the pending list. Thus, at step 6, (p_6, set_Root) is the only choice. The labelling^{dep} function sets w_6, "前行" as the head of the sentence. As all partial analysis are marked as processed, the parsing process stops. Head-modifier relationships can be directly read-off the dependency tree. That is, parent is the head of its children.

```
1   Set nProcessed ← 0
2   p_1, ... p_n ← w_1, ... w_n
3   Set each p_i as unprocessed
4   repeat
5     (p̄, ā) ← argmax_{p∈Pending, a∈ACT} score(p, a)
6     labelling^dep(p̄, ā)
7     set p̄ as processed,
8     remove p̄ from pending
9     nProcessed ← nProcessed + 1
10  until nProcessed = |T|
```

FIGURE 2 – Main loop of easy-first parsing algorithm

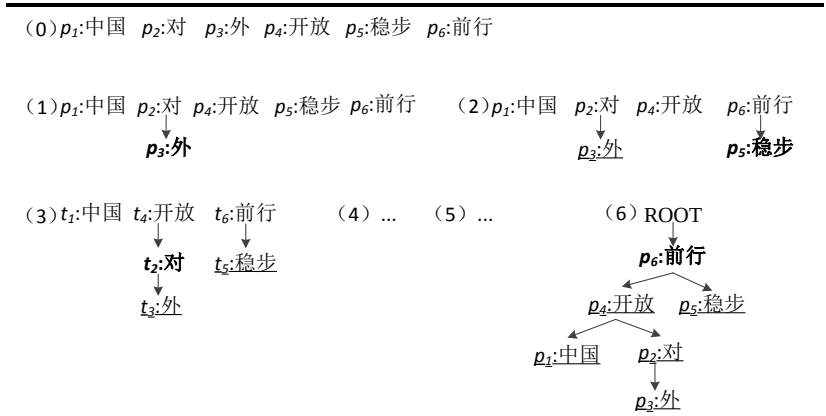

FIGURE 3 – A trace of easy-first parser for sentence "中国 对 外 开放 稳步 前行" ("China's opening up steadily advances")

2.4 The proposed easy-first joint POS tagging and dependency parsing method

In this section, we describe the joint tagging and dependency parsing method which is under the easy-first framework. Before we go into the details about the algorithm, one thing should be mentioned. That is, for dependency parsing, POS tag serves as indispensable information (Li et al., 2010). On the one hand, parsing barely based on bi-lexical features without POS tags hurt performance dramatically. On the other hand, if we tag the whole sentence first and then do dependency parsing, then syntactic features which are demonstrated to be effective for tagging disambiguation cannot be utilized. In order to provide POS tags for syntactic disambiguation and provide syntactic features for tagging disambiguation, we add a simple constraint to control the inference order: dependency relationships can only be extracted between two tagged words. Under such constraint, we can guarantee that at least POS tags for the pair of words from which dependency relation is to be extracted are already available. Also, syntactic feature can be utilize-

```
1   Set nProcessed ← 0,
2   p_1, ... p_n ← w_1, ... w_n
3   Set each p_i as unprocessed
4   repeat
5     Initialize valid
6     (p̄, l̄) ← argmax_{(p,l)∈valid} score(p, l)
7     labelling^Joint(p̄, l̄)
8     if (l̄ ∈ ACT)
9       Set p̄ as processed,
10        remove p̄ from pending
11      nProcessed ← nProcessed + 1
12  until nProcessed = |T|
```

FIGURE 4 – Main part of the joint tagging and dependency parsing algorithm

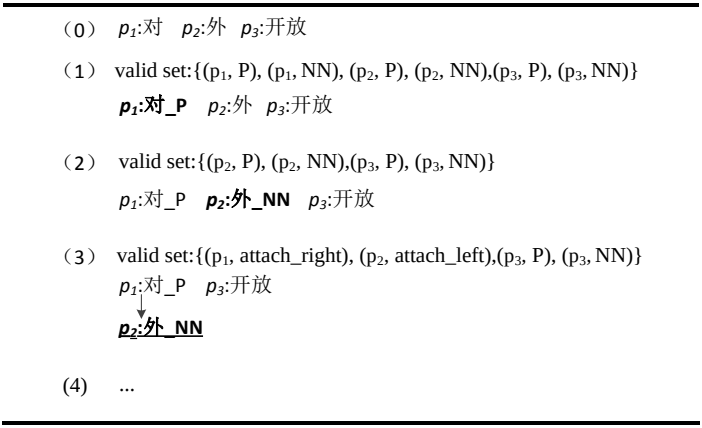

FIGURE 5 – A trace of easy-first joint tagging and parsing for "对 外 开放"

-d to process the untagged words. Under such constraint, the joint method can be constructed by a simple combination of easy-first POS tagging and easy-first dependency parsing.

Similar to previous tasks, for the joint task, t_i still corresponds to the i-th word of the input sentence w_i. Pending list is also used to record the partial dependency tree structures. The label set of the joint task is the union of **POS** and **ACT**. The labelling function of the joint task, $\text{labelling}^{\text{Joint}}$, behaves differently on the two types of labels. Particularly, $\text{labelling}^{\text{Joint}}$ (p_i, x) calls $\text{labelling}^{\text{POS}}$ (w_i, x) which associates w_i, the root of p_i, with POS tag x. $\text{labelling}^{\text{Joint}}$ (p_i, a) calls $\text{labelling}^{\text{Dep}}$ (p_i, a) which performs action a on p_i.

The main part of the joint method is shown in figure 4. Note that, to satisfy the constraint described above, at the beginning of each step, a **valid** set of (p, l) pairs are initialized (line 5). For the partial analysis p, if its root word is not yet tagged, only (p, x) are considered as valid where x is a certain POS tag. This enforces that partial analysis must be tagged first. For p which is already tagged, if its neighbour is also tagged, then attaches are allowed between the two neighbours.

After initializing the **valid** set, the joint algorithm computes scores of the (p, l) pairs from the **valid** set. Then the one with the highest score is selected and be fed to $\text{labelling}^{\text{Joint}}$. A partial analysis p is marked as processed only when p is attached to its neighbours or be set as the root of the sentence (line 8 to line 11). Figure 5 gives a toy example of the joint tagging and parsing procedure. The **valid** set of each step is also shown in the figure. For simplicity, we suppose that the POS tag set only contains two tags NN (noun) and P (preposition).

In summary, the key idea behind easy-first method is to always choose the easy items to process. The degree of easy or hard is determined by the scoring function. In the next section, we show how to learn the scoring function from training data.

3 Training

The scoring functions used in this work are all based on a linear model.

$$score(t, l) = \vec{w} \cdot \varphi(t, l)$$

Here, $\vec{w}$ is the model's parameter or weight vector and $\varphi(t, l)$ is the feature vector extracted from (t, l) pair. In this section, we first describe a general training algorithm which can be used to train models for all the tasks mentioned above and then introduce our training method which is specially designed for the joint model.

3.1 Easy-first training

The generalized easy-first training algorithm is shown in algorithm 2 which is based on the structured perceptron. Starting from a zero weight vector, the training process makes several iterations through the training set. Each iteration, algorithm 2 is utilized to process the training instances. A training instance consists of a sequence of items T together with its gold reference *R.* Here, R takes different forms in different tasks. For example, in POS tagging, R is the gold tag sequence of the input sentence. In dependency parsing, R is the gold dependency tree of the input sentence.

When $(\bar{t}, \bar{l})$ is not compatible[3] with R, the training algorithm updates the current parameter by punishing the features fired in the chosen $(\bar{t}, \bar{l})$ pair and rewarding the features fired in the (t, l) pair which is compatible with R. The algorithm will then move on to the next instance. Note that there might be more than one such pair that are compatible with R. For example, for figure 1 step (2), (w_2, P), (w_3, NN), (w_4, NN), (w_5, AD) and (w_6, VV) are all compatible with the gold tag sequence. Thus, at the beginning of each step, a **compatible** set of (*t, l*) pairs are initialized and

Algorithm 2: Easy-First Training

Input T: $t_1, \ldots, t_n$, R: gold reference of T, $labelling$: task specific function L:{ $l_1, \ldots, l_n$}, $\vec{w}$: feature weight vector, isAllowed: task specific function

Output $\vec{w}$: feature weight vector

$nProcessed \leftarrow 0$
set each t_i as unprocessed
repeat
  $\boldsymbol{compatible} \leftarrow \{(t, l) | isAllowed(t, l, R)_{\,t.state=unprocessed, l \in L} = true\}$
  $(\bar{t}, \bar{l}) \leftarrow \text{argmax}_{l \in L, t.state=unprocessed}\, score(t, l)$
  if $(\bar{t}, \bar{l}) \in \boldsymbol{compatible}$
    $labelling(\bar{t}, \bar{l})$,
    $\bar{t}.state \leftarrow processed$,
    $nProcessed \leftarrow nProcessed + 1$
  else
    $(\hat{t}, \hat{l}) \leftarrow \text{argmax}_{(t,l) \in \boldsymbol{goldAllowed}}\, score(t, l)$
    $\vec{w} \leftarrow \vec{w} + \varphi(\hat{t}, \hat{l}), \quad \vec{w} \leftarrow \vec{w} - \varphi(\bar{t}, \bar{l})$
    break
until $nProcessed = |T|$
return $\vec{w}$

[3] A (*t, l*) pair is compatible with *R* means that in *R*, *t* is labelled with *l*.

the one with the highest score is chosen to boost (Goldberg and Elhadad, 2010). Function isAllowed is used to check whether a candidate (t, l) pair belongs to the **compatible** set (line 4). Similar to the labelling function, isAllowed can be easily instantiated for different tasks. For POS tagging, $\text{isAllowed}^{\text{POS}}$ (w_i, x, R) checks whether x is the gold tag of w_i. For dependency parsing, $\text{isAllowed}^{\text{Dep}}$ (p_i, a, R) checks:

- Whether all p_i's children supposed by the gold dependency tree has already attached to p_i.
- Whether action a attaches p_i to the correct partial analysis.

For joint POS tagging and dependency parsing, $\text{isAllowed}^{\text{Joint}}$ behaves differently according to the types of labels:

- $\text{isAllowed}^{\text{Joint}}$ (p_i, x, R) returns $\text{isAllowed}^{\text{POS}}$ (w_i, x, R).
- $\text{isAllowed}^{\text{Joint}}$ (p_i, a, R) returns $\text{isAllowed}^{\text{Dep}}$ (p_i, a, R).

3.2 Training with additional loss

Algorithm 2 is a variant of the structured perceptron algorithm. The objective is to minimize the loss $\ell(R, \bar{R})$, usually hamming loss (Daumé III et al., 2009), between gold reference R and the predicted output $\bar{R}$. Whenever there is a positive loss, parameters are updated. For POS tagging, algorithm 2 minimizes $\ell^{POS}(R^{POS}, \bar{R}^{POS})$ where R^{POS} and $\bar{R}^{POS}$ are predicted and gold tag sequence respectively. For dependency parsing, algorithm 2 minimizes $\ell^{Dep}(R^{Dep}, \bar{R}^{Dep})$ where R^{Dep} and $\bar{R}^{Dep}$ are predicted and gold dependency tree, respectively.

For the joint task, algorithm 2 minimizes $\ell^{Joint}(< R^{POS}, R^{Dep} >, < \bar{R}^{POS}, \bar{R}^{Dep} >)$ where $< R^{POS}, R^{Dep} >$ denotes the pair of predicted tag sequence and dependency tree. Such loss has no distinction between tagging and parsing: either tagging error or parsing error will lead to non-zero loss and parameters are updated. This loss may not optimal in terms of achieving the best result of the individual tasks.

Inspired by (Hall et al., 2011), we slightly modify algorithm 2 to incorporate additional loss which only relates to one of the individual tasks so as to train the joint model towards the that task. Algorithm 3 shows the pseudo-code. The basic idea is that at each iteration of the training process, one of the three losses $\{\ell^{POS}, \ell^{Dep}, \ell^{Joint}\}$ is selected and parameters are updated acco-

Algorithm 3: Training with additional loss for joint POS tagging and parsing

Input T: $w_1, \ldots, w_n$, R:gold reference of T, $labelling^{Joint}$, $\boldsymbol{L^{Joint}}$, $\vec{w}$, $\text{isAllowed}^{\text{Joint}}$, ℓ: which can be either ℓ^{POS} or ℓ^{Dep} or ℓ^{Joint}.

Output $\vec{w}$: feature weight vector

1 Set $nProcessed \leftarrow 0$, $p_1, \ldots p_n \leftarrow w_1, \ldots w_n$
2 Set each p_i as unprocessed
3 **repeat**
4 Initialize $\boldsymbol{valid}$
5 $\boldsymbol{compatible} \leftarrow \{(p,l) | isAllowed^{Joint}(p,l)_{(p,l)\in \boldsymbol{valid}} = true\}$
6 **if** $(\ell = \ell^{POS})$
7 $\boldsymbol{compatible} \leftarrow \boldsymbol{compatible} \cup \{(p,l) | (p,l) \in \boldsymbol{valid} \wedge l \in \boldsymbol{ACT}\}$
8 **if** $(\ell = \ell^{Dep})$
9 $\boldsymbol{compatible} \leftarrow \boldsymbol{compatible} \cup \{(p,l) | (p,l) \in \boldsymbol{valid} \wedge l \in \boldsymbol{POS}\}$
10 $(\bar{p}, \bar{l}) \leftarrow \text{argmax}_{(p,l)\in \boldsymbol{valid}} score(p,l)$
11 **if** $(\bar{p}, \bar{l}) \in \boldsymbol{compatible}$
12 ... (*the following are exactly the same as algorithm 2*)

-rdingly: if ℓ^{Joint} is selected, then both tagging and parsing error cause parameter update; if ℓ^{POS} is selected, then only tagging errors can cause parameter update. This can be done by adding all valid attach actions to the **compatible** set regardless whether those actions are indeed compatible with the gold reference (line 6 to line 7); For ℓ^{Dep}, only parsing errors cause parameter update which can be achieved similar to the case of ℓ^{POS}.

4 Experiments

To make comparison with previous works, we use Penn Chinese Treebank 5.1 (CTB5) (Xue et al., 2005) to evaluate our method. We use the standard split of CTB5 as described in (Duan et al., 2007): section 001-815 and 1001-1136 are used as training set, section 886-931 and 1148-1151 are used as development set, section 816-885 and 1137-1147 are used as test set. Head finding rules of (Zhang and Clark 2008b) are used to convert the constituent trees into dependency trees. An Intel Core i7 870 2.93 GHz machine is used for evaluation. For POS tagging and dependency parsing, the number of training iterations are selected according to the model's performance on the development set. The model which achieves the highest score on the development set is selected to run on the test set.

4.1 POS tagging

Following Zhang and Clark (2008a), in the experiments, we also use a dictionary to limit the number of candidate tags for each word. That is, for a word which occurs more than M times in the training data, the tagger only considers the POS tags co-occurred with that word in the training data. We found 6 to be the optimal value on the development set. The feature templates we used in the experiments are shown in table 1. Z&C08 denotes the features templates of (Zhang and Clark, 2008a) which include uni-gram and bi-gram as well as some character based features. In addition to Z&C08, we also incorporate bi-directional POS tag features and the resulted feature set is denoted by feature set BD. For feature set OP, we include some order preference features with the goal to signal to the tagger that some words should be delayed to late stages until more surrounding tags are available and more informed prediction can be made.

Table 2 shows the performance for different feature set and also gives state-of-the-art results on the same data set. "Li-10" denotes the performance of the tagger of (Li et al., 2010) and Hatori-11 denotes the performance of the tagger of (Hatori et al., 2011). From table 2, we can see

Feature sets	Tagging Feature Templates
Z&C08	w_i, w_{i+1}, w_iw_{i+1}, w_iw_{i-1}, $FC(w_i)$, $LC(w_i)$, $TS(FC(w_i))$, $TS(LC(w_i))$, $C_n(w_i)$ (n=2…len(w_i) -1), $LC(w_{i-1})w_iFC(w_{i+1})$, $FC(w_i)C_n(w_i)$ (n=2…len(w_i)), $LC(w_i)C_n(w_i)$ (n=1…len(w_i) -1), $C_n(w_i)$ (if $C_n(w_i) = C_{n+1}(w_i)$), tag_{i-1}, $tag_{i-1}tag_{i-2}$
BD	Z&C08 + tag_{i+1}, $tag_{i+1}tag_{i+2}$, $tag_{i-1}tag_{i+1}$
OP^{POS}	BD + $w_itagged_{i+1}$, $w_itagged_{i-1}$, $w_itagged_{i-2}$, $w_itagged_{i+2}$ $w_itagged_{i-2}tagged_{i-1}$, $w_itagged_{i-1}tagged_{i+1}$, $w_itagged_{i+1}tagged_{i+2}$

TABLE 1 – Feature templates for easy-first POS tagging. Here w_i denotes the i-th word of the input sentence, tag_i denotes the POS tag of w_i, $FC(w_i)$/$LC(w_i)$ denotes the first/last character of w_i. $C_n(w_i)$ is the n-th character of w_i, TS(c) is the POS tag set that co-occurred with character c in the dictionary. $tagged_k$ denotes whether w_k has already been tagged

Feature Sets	Development Set			Test Set			Speed[4]
	Total	Seen	Unseen	Total	Seen	Unseen	
Z&C08	93.58	94.59	**77.16**	93.36	94.22	80.49	3001
BD	93.85	94.98	75.65	93.68	94.53	**80.83**	2733
OP	94.05	**95.13**	76.65	**93.84**	**94.73**	80.49	2070
Li-10	–	–	–	93.51	94.36	80.78	292
Hatori-11	**94.15**	–	–	93.82	–	–	210

TABLE 2 – POS tagging performance

that bi-directional features are effective for improving tagging performance. With bi-directional features, the tagger achieves the accuracy of 93.68 which is higher than Li-10 and slightly lower than Hatori-11. By incorporating order preference features, tagging accuracy increases to 93.84 better than both Li-10 and Hatori-11. Moreover, rather than using Viterbi or beam search, our easy-first tagger is deterministic which is easy to implement and runs in high speed, more than 2000 sentence per second.

4.2 Dependency parsing

We use root accuracy, complete match rate and word accuracy or dependency accuracy to evaluate the parser's performance. Feature templates for easy-first dependency parsing are shown in table 3. G&E10 denotes the feature templates used in (Goldberg and Elhadad, 2010) with some modification: the feature templates in the last row of G&E10 were originally designed to deal with English PP attachment ambiguity. Those templates are limited to be used only when

Feature sets	Parsing Feature Templates
G&E10	for p in p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}, p_{i+3}: len(p), nc(p)
	for p q in(p_{i-2}, p_{i-1}),(p_{i-1}, p_i),(p_i, p_{i+1}),(p_{i+1}, p_{i+2}),(p_{i+2}, p_{i+3}): dis(p, q), dis(p, q)$tag_p tag_q$
	for p in p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}, p_{i+3}: tag_p, w_p, $tag_p lc_p$, $tag_p rc_p$, $tag_p lc_p rc_p$
	for p q in(p_i, p_{i+2}),(p_{i-1}, p_i),(p_i, p_{i+1}),(p_{i-1}, p_{i+2}),(p_{i+1}, p_{i+2}): $tag_p tag_q$, $tag_p tag_q lc_p lc_q$, $w_p w_q$, $tag_p w_q$, $w_p tag_q$, $tag_p tag_q lc_p rc_q$, $tag_p tag_q rc_p lc_q$, $tag_p tag_q rc_p rc_q$
	$w_{pi-1} w_{pi} rc_{pi}$, $tag_{pi-1} w_{pi} rcw_{pi}$, $w_{pi-1} w_{pi+1} rc_{pi+1}$, $tag_{pi-1} w_{pi+1} rcw_{pi+1}$, $w_{pi} w_{pi+1} rc_{pi+1}$ $tag_{pi} w_{pi+1} rcw_{pi+1}$, $w_{pi+1} w_{pi+2} rc_{pi+2}$, $tag_{pi+1} w_{pi+2} rcw_{pi+2}$, $w_{pi} w_{pi+2} rc_{pi+2}$, $tag_{pi+1} w_{pi+2} rcw_{pi+2}$
VTT^{DEP}	for p in p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}: $w_p$vl(p) , $w_p$vr(p), tag_pvl(p), tag_pvr(p), $r_2 cw_p$, $r_2 c_p$, $l_2 cw_p$, $l_2 c_p$, $tag_p lc_p$ rc_p
	$tag_{pi-2} tag_{pi-1} tag_p$, $tag_{pi-1} tag_{pi} tag_{p+1}$, $tag_{pi} tag_{pi+1} tag_{p+2}$
OP^{DEP}	for p, q in (p_i, p_{i-1}),(p_i, p_{i-2}),(p_i, p_{i+1}),(p_i, p_{i+2}),(p_i, p_{i+3}):$w_p$nc(q), tag_pnc(q), $tag_p tag_q$nc(q),
	for p, q, r in (p_i, p_{i-1}, p_{i-2}),(p_i, p_{i-1}, p_{i+1}),(p_i, p_{i+1}, p_{i+2}): $w_p$nc(q) nc(r), tag_pnc(q) nc(r)

TABLE 3 – Feature template used for dependency parsing. For a partial dependency tree p, len(p) is the number of words in p. nc(p) denotes whether p is a leaf node. w_p and tag_p denote p's root word and the POS tag of p's root word, respectively. lc_p/rc_p denote the POS tag of p's left/right most child. lcw_p/rcw_p denotes the word form of p's left/right most child. $l_2 c_p$/$r_2 c_p$ denote the POS tag of p's second left/right most child. $l_2 cw_p$/$r_2 cw_p$ denotes the word form of p's second left/right most child

[4] Since experiments in this work and that in the other work are carried on different machines, speed is for reference only.

		H&S	Z&N	H&S-H	Z&N-H	Li-O2	Li-O3	G&E10	VTT	OP
GoldPOS	Word	85.20	86.00	85.12	85.96	**86.18**	86.00	84.62	85.18	85.22
	Root	78.32	–	78.30	**80.87**	78.58	77.59	74.70	75.38	75.48
	Compl	33.72	**36.90**	32.77	35.03	34.07	34.02	36.12	36.27	36.80
Tag Accuracy		–	–	93.82		93.51		93.84		
AutoPOS	Word	–	–	77.13	**78.04**	–	–	77.45	77.64	77.66
	Root	–	–	72.49	**75.55**	–	–	68.50	68.92	68.35
	Compl	–	–	25.13	26.07	–	–	**28.89**	28.19	28.45
J-N	Word	–	–	–	–	79.03	**79.29**	78.43	78.73	78.87
	Root	–	–	–	–	**74.70**	74.65	67.14	68.29	68.50
	Compl	–	–	–	–	27.19	27.24	28.98	**29.34**	29.29
Speed		–	–	32.7	9	5.8	2	**391**	385	355

TABLE 4 – Parsing performance. H&S-10 and Z&N-11 denote parsers in Huang and Sagae (2010) and Zhang and Nirve (2011), respectively. H&S-H and Z&N-H denote Hatori et al., (2011)'s re-implementation of H&S-10 and Z&N-11, respectively. Li-10-O2/O3 denotes the 2rd/3rd graph based model of Li et al., (2010)

either tag_{p_i}or $tag_{p_{i+1}}$or $tag_{p_{i+2}}$is a preposition. For Chinese, PP attachment ambiguity is not as prevalent as that of English (Huang et al., 2009) and we found that use these features without any limitation yields better results. VTT includes valence features, tri-gram features and third order features which were proved useful for transition based parsers (Zhang and Nivre, 2011). For OP, some additional order preference feature templates are added.

Parsing results are shown in table 4. "GoldPOS" denotes the input with gold standard POS tag. "AutoPOS" denotes that the training set are assigned with gold standard POS tag while the test set are tagged by our easy-first tagger. "J-N" denotes that we use 10-fold Jack-Nifing to train the model. "Tag Accuracy" denotes the test set tagging accuracy. From table 4, we can see that valence and tri-gram features are also effective for easy-first parser. For GoldPOS, word accuracy boosted from 84.62 to 85.18. For AutoPOS and J-N, word accuracy also increased about 0.2 and 0.3 point, respectively. Order preference features are not as effective as it were for tagging. After adding these features, parsing performance rarely changed. One reason might be that some of the features of G&E10 already capture order information and the order preference features list in table 3 redundant to some degree. Comparing with other state-of-the-art parsers on this data set, the performance of the easy-first parser is still lower. This may due to the fact that our parser is greedy, thus more vulnerable to error propagation. Interestingly, for AutoPOS and J-N, the easy-first parser achieves the highest complete match rate. This is consistent with (Goldberg and Elhadad, 2010). One thing should be mentioned is that, the deterministic easy-first parser is both easy to implement and runs very fast, more than 350 sentences per second.

4.3 Joint tagging and parsing

Similar to (Hatori et al., 2011), for the joint task, we choose the model which performs the best on the development set in terms of word accuracy to run on the test set. Feature templates for joint POS tagging and dependency parsing are shown in table 5. The feature set is the union of the feature set used for POS tagging and the feature set used for dependency parsing. Besides, in-

Feature sets	Feature templates for Joint tagging and parsing
Syn	nc(p_{i-1}) $w_{p_{i-1}}$, nc(p_{i-1}) $tag_{p_{i-1}}$ nc(p_{i+1}) $w_{p_{i+1}}$, nc(p_{i+1}) $tag_{p_{i+1}}$ nc(p_{i-1})w_i $tag_{p_{i-1}}$, nc(p_{i-1})$w_i w_{p_{i-1}}$, nc(p_{i-1}) $tag_{p_{i-1}} lc_{p_{i-1}}$ nc(p_{i-1}) $tag_{p_{i-1}} rc_{p_{i-1}}$ nc(p_{i+1})$w_i tag_{p_{i+1}}$ nc(p_{i+1})$w_i w_{p_{i+1}}$ nc(p_{i+1}) $tag_{p_{i+1}} lc_{p_{i+1}}$ nc(p_{i+1}) $tag_{p_{i+1}} rc_{p_{i+1}}$
VVT^{Joint}	Syn + VVT^{DEP} + OP^{POS}

Table 5 Feature templates for joint POS tagging and dependency parsing.

-spired by (Hatori et al., 2011), we also incorporate some syntactic features that aim to improve tagging accuracy and theses features are denoted by Syn.

Table 6 shows the results for the joint model with different losses. Parsing performance, especially word level accuracy, is largely affected by the different loss settings. When training the joint model with a single loss ℓ^{Joint}, word level accuracy is 79.12. When training with $\ell^{Joint}$and $\ell^{Parsing}$, word level accuracy increased to 79.91. These results demonstrate that our training method can bias the joint model towards the desired task. However, as we try different losses, tagging accuracy rarely changes. This may because that the tagging accuracy is already very high and it is quite difficult to achieve further improvement.

Comparing with previous results on joint POS tagging and dependency parsing, our method achieves the best result in terms of complete match rate and POS tagging accuracy[5]. The word accuracy is still below the best result. This may due to the fact that our joint decoder is deterministic thus suffers more from error propagation comparing with beam search based or dynamic programming based decoders. However, our joint method can also be enhanced with beam search and we leave it to future work.

Model	losses	POS	Word	Root	Compl	Speed
Joint	ℓ^{Joint}	94.25*	79.12	72.02	30.66	70+
	$\ell^{Joint}, \ell^{Parsing}$	94.26*	79.91*	72.81	**30.76**	70+
	ℓ^{Joint}, ℓ^{POS}	**94.27***	79.04	71.44	30.29	70+
Pipeline	–	93.84	78.73	68.29	29.34	

Other Methods	POS	Word	Root	Compl	Speed
B&N-12	93.24	81.42	–	–	–
Li-10-V1-O2	93.08	80.74	75.80	28.24	1.7
Li-10-V2-O3	92.80	80.79	75.84	29.11	0.3
Z&N-Hatori	93.94	**81.33**	**77.92**	29.90	1.5
H&S-Hatori	94.01	79.83	73.86	27.85	9.5

TABLE 6 – Joint tagging and parsing results. B&N-12 denotes the result of (Bohnet and Nivre, 2012). Li-10-V1-O2 and Li-10-V2-O3 denote results from (Li et al., 2010). Z&N-Hatori and H&S-Hatori denote results from (Hatori et al., 2011). "*" denotes statistical significant ($p < 0.05$ by MeNemar's test) comparing with the pipeline method.

[5] Comparing with easy-first tagger, the joint model does better at disambiguiting NN-VV, DEC-DEG while poorer at JJ-NN. This result is similar to previous result (Hatori et al, 2011). For limited space, we omit the confusion matrix.

5 Related Work

The easy-first dependency parsing algorithm was first proposed by (Goldberg and Elhadad, 2010), they applied the algorithm to English dependency parsing and by combining results with transition based parser and graph based parser, state-of-the-art performance is achieved. This work is a generalization to their algorithm, and we applied the generalized algorithm to Chinese POS tagging and dependency parsing and also achieves good results in terms of both speed and accuracy. We also proposed the first easy-first joint tagging and parsing algorithm. By incorporating additional loss during training, our method achieves the best tagging accuracy reported so far. Shen et al. (2007) proposed a bi-directional POS tagging algorithm which achieves state-of-the-art accuracy on English POS tagging. Comparing to their method, our tagging algorithm in this paper is much simpler and we are the first to use order preference features in POS tagging. Also this is the first work that applies easy-first tagging on Chinese.

For joint POS tagging and (unlabelled, projective) dependency parsing, Li et al. (2010) proposed the first graph based algorithm. Lee et al. (2011) proposed a graphical model to solve the joint problem. Hatori et al. (2011) proposed the first transition based algorithm. Bohnet and Nivre (2012) extended Hatori et al. (2011) to labelled non-projective dependency parsing. Different from the works talked above, our method is based on the easy-first framework. In addition, all previous joint methods optimize a single loss in the training phase while we are the first to train the joint model with additional loss.

Hall et al. (2011) proposed the augmented-loss training for dependency parser that aims at adapting the parser to other domains or to downstream tasks such as MT reordering. They extended structured perceptron with multiple losses each of which is associated with an external training set. Our method is directly inspired by Hall et al. (2011). However, rather than domain adaptation, our method aims at training the joint model to pursue the best result of one of the individual task. Moreover, our method optimizes all loss on a single training set.

6 Conclusion

In this paper, we generalize the method of (Goldberg and Elhadad, 2010) to a general framework and apply the framework to Chinese POS tagging and dependency parsing. We also proposed the first joint tagging and dependency parsing algorithm under the easy-first framework. We show that by using order preference features, an easy-first POS tagger can achieve the state-of-the-art accuracy. We show a deterministic easy-first parser can surpass the transition-based parser when the input is associated with automatically generated tags. We also illustrate that by incorporating additional loss in the training process, we can bias the joint model towards the desired task.

Acknowledgments

We would like to thank Mu Li, ChangNing Huang, Yova Goldberg, Nan Yang, Zhanghua Li and Jun Hatori for frequent discussions. We also thank Muhua Zhu, Wenliang Chen, Nan Yang and three anonymous reviewers for their insightful suggestions on earlier draft of this paper. This work was supported in part by the National Science Foundation of China (61100089, 61073140, 61272376, 61003159), specialized Research Fund for the Doctoral Program of Higher Education (20100042110031) and the Fundamental Research Funds for the Central Universities (N110404012).

References

Bohnet, B. and Nivre J. (2012) A Transition-Based System for Joint Part-of-Speech Tagging and Labeled Non-Projective Dependency Parsing. (EMNLP 2012).

Collins, M. (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms.(EMNLP 2002), pages 1-8

Daumé III, H., Langford, J. and Marcu, D. (2009) Search-based structured prediction. In Journel of Machine Learning, 75(3): 297-325.

Duan, X., Zhao, J. and Xu, B. (2007) Probabilistic parsing action models for multilingual dependency parsing. (EMNLP-CoNLL 2007)

Goldberg, Y. and Elhadad, M. (2010) An Efficient Algorithm for Eash-First Non-Directional Dependency Parsing. (NAACL 2010), pages

Hall, K., McDonald, R., Katz-Brown, J. and Ringgaard, M. (2011) Training dependency parsers by jointly optimizing multiple objectives. (EMNLP 2011)

Hatori, J., Matsuzaki, T., Miyao, Y. and Tsujii, J. (2011) Incremental Joint POS Tagging and Dependency Parsing in Chinese. (IJCNLP 2011), pages 1216-1224, Chiang Mai, Thailand.

Huang, L. Jiang, W. and Liu, Q. (2009) Bilingually-Constrained (Monolingual) Shift-Reduce Parsing. (EMNLP 2009), pages 1222-1231, Singapore.

Huang, L. and Sagae, K. (2010) Dynamic programming for linear-time incremental parsing. (ACL 2010).

Lee, J., Naradowsky, J. and Smith, D.A. (2011) A discriminative model for joint morphological disambiguation and dependency parsing. (ACL 2011)

Li, Z., Zhang, M., Che, W., Liu, T. Chen, W. and Li, H. (2010) Joint Models for Chinese POS Tagging and Dependency Parsing (EMNLP 2010), pages 1180-1191, Edinburgh.

Rataparkhi, A. (1996) A Maximum Entropy Part-Of-Speech Tagger. (EMNLP 1996)

Shen, L., Satt, G. and Joshi, A. K. (2007) Guided Learning for Bidirectional Sequence Classification. (ACL 2007), pages 760-767, Prague.

Tsuruoka, Y. Tsujii, J. (2005). Bidirectional inference with the easiest-first strategy for tagging sequence data. (EMNLP 2005) pages 467-474.

Xue, N., Xia, F., Chiou, F. and Palmer, M. (2005) The Penn Chinese Treebank: Phrase structure annotation of a large corpus. In Natural Language Engineering, volume 11, pages 207-238.

Zhang, Y. and Clark, S. (2008a) Joint Word Segmentation and POS Tagging Using a Single Perceptron. (ACL 2008), Ohia.

Zhang, Y. and Clark, S. (2008b) A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. (EMNLP 2008), Hawaii.

Zhang, Y. and Nivre, J. (2011) Transition-based Dependency Parsing with Rich Non-local Features. (ACL 2011), Portland.