Exploring Deterministic Constraints: From a Constrained English POS Tagger to an Efficient ILP Solution to Chinese Word Segmentation

Qiuye Zhao **Mitch Marcus**
Dept. of Computer & Information Science
University of Pennsylvania
qiuye, mitch@cis.upenn.edu

Abstract

We show for both English POS tagging and Chinese word segmentation that with proper representation, large number of deterministic constraints can be learned from training examples, and these are useful in constraining probabilistic inference. For tagging, learned constraints are directly used to constrain Viterbi decoding. For segmentation, character-based tagging constraints can be learned with the same templates. However, they are better applied to a word-based model, thus an integer linear programming (ILP) formulation is proposed. For both problems, the corresponding constrained solutions have advantages in both efficiency and accuracy.

1 introduction

In recent work, interesting results are reported for applications of integer linear programming (ILP) such as semantic role labeling (SRL) (Roth and Yih, 2005), dependency parsing (Martins et al., 2009) and so on. In an ILP formulation, 'non-local' deterministic constraints on output structures can be naturally incorporated, such as "*a verb cannot take two subject arguments*" for SRL, and the projectivity constraint for dependency parsing. In contrast to probabilistic constraints that are estimated from training examples, this type of constraint is usually hand-written reflecting one's linguistic knowledge.

Dynamic programming techniques based on Markov assumptions, such as Viterbi decoding, cannot handle those 'non-local' constraints as discussed above. However, it is possible to constrain Viterbi decoding by 'local' constraints, e.g. "*assign label t to word w*" for POS tagging. This type of constraint may come from human input solicited in interactive inference procedure (Kristjansson et al., 2004).

In this work, we explore deterministic constraints for two fundamental NLP problems, English POS tagging and Chinese word segmentation. We show by experiments that, with proper representation, large number of deterministic constraints can be learned automatically from training data, which can then be used to constrain probabilistic inference.

For POS tagging, the learned constraints are directly used to constrain Viterbi decoding. The corresponding constrained tagger is 10 times faster than searching in a raw space pruned with beam-width 5. Tagging accuracy is moderately improved as well.

For Chinese word segmentation (CWS), which can be formulated as character tagging, analogous constraints can be learned with the same templates as English POS tagging. High-quality constraints can be learned with respect to a special tagset, however, with this tagset, the best segmentation accuracy is hard to achieve. Therefore, these character-based constraints are not directly used for determining predictions as in English POS tagging. We propose an ILP formulation of the CWS problem. By adopting this ILP formulation, segmentation F-measure is increased from 0.968 to 0.974, as compared to Viterbi decoding with the same feature set. Moreover, the learned constraints can be applied to reduce the number of possible words over a character sequence, i.e. to reduce the number of variables to set. This reduction of problem size immediately speeds up an ILP solver by more than 100 times.

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 1054–1062,
Jeju, Republic of Korea, 8-14 July 2012.

2 English POS tagging

2.1 Explore deterministic constraints

Suppose that, following (Chomsky, 1970), we distinguish major lexical categories (Noun, Verb, Adjective and Preposition) by two binary features: $+|-$ N and $+|-$ V. Let (+N, −V) =Noun, (−N, +V) =Verb, (+N, +V) =Adjective, and (−N, −V) =preposition. A word occurring in between a preceding word *the* and a following word *of* always bears the feature +N. On the other hand, consider the annotation guideline of English Treebank (Marcus et al., 1993) instead. Part-of-speech (POS) tags are used to categorize words, for example, the POS tag VBG tags verbal gerunds, NNS tags nominal plurals, DT tags determiners and so on. Following this POS representation, there are as many as 10 possible POS tags that may occur in between *the–of*, as estimated from the WSJ corpus of Penn Treebank.

2.1.1 Templates of deterministic constraints

To explore determinacy in the distribution of POS tags in Penn Treebank, we need to consider that a POS tag marks the basic syntactic category of a word as well as its morphological inflection. A constraint that may determine the POS category should reflect both the context and the morphological feature of the corresponding word.

The practical difficulty in representing such deterministic constraints is that we do not have a perfect mechanism to analyze morphological features of a word. Endings or prefixes of English words do not deterministically mark their morphological inflections. We propose to compute the morph feature of a word as the set of all of its possible tags, i.e. all tag types that are assigned to the word in training data. Furthermore, we approximate unknown words in testing data by rare words in training data. For a word that occurs less than 5 times in the training corpus, we compute its morph feature as its last two characters, which is also conjoined with binary features indicating whether the rare word contains digits, hyphens or upper-case characters respectively. See examples of morph features in Table 1.

We consider *bigram* and *trigram* templates for generating potentially deterministic constraints. Let w_i denote the i_{th} word relative to the current word w_0; and m_i denote the morph feature of w_i. A *bigram* constraint includes one contextual word $(w_{-1}|w_1)$ or the corresponding morph feature; and a *trigram* constraint includes both contextual words or their morph features. Each constraint is also conjoined with w_0 or m_0, as described in Table 2.

(frequent)	(set of possible tags of the word)
w_0=*trades*	m_0={NNS, VBZ}
(rare)	(the last two characters...)
w_0=*time-shares*	m_0={-es, HYPHEN}

Table 1: Morph features of frequent words and rare words as computed from the WSJ Corpus of Penn Treebank.

bi-	$w_{-1}w_0, w_0w_1, m_{-1}w_0, w_0m_1$
-gram	$w_{-1}m_0, m_0w_1, m_{-1}m_0, m_0m_1$
tri-	$w_{-1}w_0w_1, m_{-1}w_0w_1, w_{-1}m_0w_1, m_{-1}m_0w_1$
-gram	$w_{-1}w_0m_1, m_{-1}w_0m_1, w_{-1}m_0m_1, m_{-1}m_0m_1$

Table 2: The templates for generating potentially deterministic constraints of English POS tagging.

2.1.2 Learning of deterministic constraints

In the above section, we explore templates for potentially deterministic constraints that may determine POS category. With respect to a training corpus, if a constraint C relative to w_0 'always' assigns a certain POS category t^* to w_0 in its context, i.e. $\frac{\texttt{count}(C \wedge t_0 = t^*)}{\texttt{count}(C)} > \texttt{thr}$, and this constraint occurs more than a cutoff number, we consider it as a deterministic constraint. The threshold thr is a real number just under 1.0 and the cutoff number is empirically set to 5 in our experiments.

2.1.3 Decoding of deterministic constraints

By the above definition, the constraint of $w_{-1} =$ *the*, $m_0 = \{\texttt{NNS}, \texttt{VBZ}\}$ and $w_1 =$ *of* is deterministic. It determines the POS category of w_0 to be NNS. There are at least two ways of decoding these constraints during POS tagging. Take the word *trades* for example, whose morph feature is {NNS, VBZ}. One alternative is that as long as *trades* occurs between *the-of*, it is tagged with NNS. The second alternative is that the tag decision is made only if all deterministic constraints relative to this occurrence of *trades* agree on the same tag. Both ways of decoding are purely rule-based and involve no probabilistic inference. In favor of a higher precision, we adopt the latter one in our experiments.

raw input	$O(nT^2)$	$n = 23$
The complex financing plan in the S&L bailout law includes...		
constrained input	$O(m_1T + m_2T^2)$	$m_1 = 2, m_2 = 1$
The/DT **complex/– financing/–** plan/NN in/IN		
the/DT **S&L/–** bailout/NN law/NN includes/VBZ ...		

Table 3: Comparison of raw input and constrained input.

2.2 Search in a constrained space

Following most previous work, we consider POS tagging as a sequence classification problem and decompose the overall sequence score over the linear structure, i.e. $\hat{\mathbf{t}} = \underset{\mathbf{t} \in \mathbf{tagGEN(w)}}{\arg\max} \sum_{i=1}^{n} score(t_i)$ where function **tagGEN** maps input sentence $\mathbf{w} = w_1...w_n$ to the set of all tag sequences that are of length n.

If a POS tagger takes raw input only, i.e. for every word, the number of possible tags is a constant T, the space of **tagGEN** is as large as T^n. On the other hand, if we decode deterministic constraints first before a probabilistic search, i.e. for some words, the number of possible tags is reduced to 1, the search space is reduced to T^m, where m is the number of (unconstrained) words that are not subject to any deterministic constraints.

Viterbi algorithm is widely used for tagging, and runs in $O(nT^2)$ when searching in an unconstrained space. On the other hand, consider searching in a constrained space. Suppose that among the m unconstrained words, m_1 of them follow a word that has been tagged by deterministic constraints and m_2 (=m-m_1) of them follow another unconstrained word. Viterbi decoder runs in $O(m_1T + m_2T^2)$ while searching in such a constrained space. The example in Table 3 shows raw and constrained input with respect to a typical input sentence.

Lookahead features

The score of tag predictions are usually computed in a high-dimensional feature space. We adopt the basic feature set used in (Ratnaparkhi, 1996) and (Collins, 2002). Moreover, when deterministic constraints have applied to contextual words of w_0, it is also possible to include some lookahead feature templates, such as:

$t_0\&t_1, t_0\&t_1\&t_2,$ and $t_{-1}\&t_0\&t1$

where t_i represents the tag of the i_{th} word relative to the current word w_0. As discussed in (Shen et al., 2007), categorical information of neighbouring words on both sides of w_0 help resolve POS ambiguity of w_0. In (Shen et al., 2007), lookahead features may be available for use during decoding since searching is bidirectional instead of left-to-right as in Viterbi decoding. In this work, deterministic constraints are decoded before the application of probabilistic models, therefore lookahead features are made available during Viterbi decoding.

3 Chinese Word Segmentation (CWS)

3.1 Word segmentation as character tagging

Considering the ambiguity problem that a Chinese character may appear in any relative position in a word and the out-of-vocabulary (OOV) problem that it is impossible to observe all words in training data, CWS is widely formulated as a character tagging problem (Xue, 2003). A character-based CWS decoder is to find the highest scoring tag sequence $\hat{\mathbf{t}}$ over the input character sequence **c**, i.e.

$$\hat{\mathbf{t}} = \underset{\mathbf{t} \in \mathbf{tagGEN(c)}}{\arg\max} \sum_{i=1}^{n} score(t_i) \ .$$

This is the same formulation as POS tagging. The Viterbi algorithm is also widely used for decoding.

The tag of each character represents its relative position in a word. Two popular tagsets include 1) `IB`: where `B` tags the beginning of a word and `I` all other positions; and 2) `BMES`: where `B`, `M` and `E` represent the beginning, middle and end of a multi-character word respectively, and `S` tags a single-character word. For example, after decoding with `BMES`, 4 consecutive characters associated with the tag sequence `BMME` compose a word. However, after decoding with `IB`, characters associated with `BIII` may compose a word if the following tag is `B` or only form part of a word if the following tag is `I`. Even though character tagging accuracy is higher with tagset `IB`, tagset `BMES` is more popular in use since better performance of the original problem CWS can be achieved by this tagset.

Character-based feature templates

We adopt the 'non-lexical-target' feature templates in (Jiang et al., 2008a). Let c_i denote the i_{th} character relative to the current character c_0 and t_0

denote the tag assigned to c_0. The following templates are used:
$c_i\&t_0$ (i=-2...2), $c_ic_{i+1}\&t_0$ (i=-2...1) and $c_{-1}c_1\&t_0$.

Character-based deterministic constraints

We can use the same templates as described in Table 2 to generate potentially deterministic constraints for CWS character tagging, except that there are no morph features computed for Chinese characters. As we will show with experimental results in Section 5.2, useful deterministic constraints for CWS can be learned with tagset IB but not with tagset BMES. It is interesting but not surprising to notice, again, that the determinacy of a problem is sensitive to its representation. Since it is hard to achieve the best segmentations with tagset IB, we propose an indirect way to use these constraints in the following section, instead of applying these constraints as straightforwardly as in English POS tagging.

3.2 Word-based word segmentation

A word-based CWS decoder finds the highest scoring segmentation sequence $\hat{\mathbf{w}}$ that is composed by the input character sequence $\mathbf{c}$, i.e.

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w} \in \mathbf{segGEN(c)}} \sum_{i=1}^{|\mathbf{w}|} score(w_i) \ .$$

where function **segGEN** maps character sequence $\mathbf{c}$ to the set of all possible segmentations of $\mathbf{c}$. For example, $\mathbf{w} = (c_1..c_{l_1})...(c_{n-l_k+1}...c_n)$ represents a segmentation of k words and the lengths of the first and last word are l_1 and l_k respectively.

In early work, rule-based models find words one by one based on heuristics such as forward maximum match (Sproat et al., 1996). Exact search is possible with a Viterbi-style algorithm, but beam-search decoding is more popular as used in (Zhang and Clark, 2007) and (Jiang et al., 2008a).

We propose an Integer Linear Programming (ILP) formulation of word segmentation, which is naturally viewed as a word-based model for CWS. Character-based deterministic constraints, as discussed in Section 3.1, can be easily applied.

3.3 ILP formulation of CWS

Given a character sequence $\mathbf{c}$=$c_1...c_n$, there are $s(=n(n+1)/2)$ possible words that are contiguous subsets of $\mathbf{c}$, i.e. $w_1, ..., w_s \subseteq \mathbf{c}$. Our goal is to find an optimal solution $\mathbf{x} = x_1...x_s$ that maximizes $\sum_{i=1}^{s} score(w_i) \cdot x_i$, subject to

$$(1) \sum_{i:c \in w_i} x_i = 1, \ \forall c \in \mathbf{c};$$
$$(2)\ x_i \in \{0, 1\}, 1 \leq i \leq s$$

The boolean value of x_i, as guaranteed by constraint (2), indicates whether w_i is selected in the segmentation solution or not. Constraint (1) requires every character to be included in exactly one selected word, thus guarantees a proper segmentation of the whole sequence. This resembles the ILP formulation of the set cover problem, though the first constraint is different. Take $n = 2$ for example, i.e. $\mathbf{c} = c_1c_2$, the set of possible words is $\{c_1, c_2, c_1c_2\}$, i.e. $s = |\mathbf{x}| = 3$. There are only two possible solutions subject to constraints (1) and (2), $\mathbf{x} = 110$ giving an output set $\{c_1, c_2\}$, or $\mathbf{x} = 001$ giving an output set $\{c_1c_2\}$.

The efficiency of solving this problem depends on the number of possible words (contiguous subsets) over a character sequence, i.e. the number of variables in $\mathbf{x}$. So as to reduce $|\mathbf{x}|$, we apply deterministic constraints predicting IB tags first, which are learned as described in Section 3.1. Possible words are generated with respect to the partially tagged character sequence. A character tagged with B always occurs at the beginning of a possible word. Table 4 illustrates the constrained and raw input with respect to a typical character sequence.

n=11	|x|	input
raw	55	法正研究从波黑撤军计划
constrained	18	法正研**/B**究从波**/B**黑撤**/B**军计**/B**划

Table 4: Comparison of raw input and constrained input.

3.4 Character- and word-based features

As studied in previous work, word-based feature templates usually include the word itself, sub-words contained in the word, contextual characters/words and so on. It has been shown that combining the use of character- and word-based features helps improve performance. However, in the character tagging formulation, word-based features are non-local.

To incorporate these non-local features and make the search tractable, various efforts have been made. For example, Jiang et al. (2008a) combine different levels of knowledge in an outside linear model of a two-layer cascaded model; Jiang et al. (2008b) uses the forest re-ranking technique (Huang, 2008); and in (Kruengkrai et al., 2009), only known words in vocabulary are included in the hybrid lattice consisting of both character- and word-level nodes.

We propose to incorporate character-based features in word-based models. Consider a character-based feature function $\phi(c, t, \mathbf{c})$ that maps a character-tag pair to a high-dimensional feature space, with respect to an input character sequence $\mathbf{c}$. For a possible word over $\mathbf{c}$ of length l , $w_i = c_{i_0}...c_{i_0+l-1}$, tag each character c_{i_j} in this word with a character-based tag t_{i_j}. Character-based features of w_i can be computed as $\{\phi(c_{i_j}, t_{i_j}, \mathbf{c})|0 \leq j < l\}$. The first row of Table 5 illustrates character-based features of a word of length 3, which is tagged with tagset `BMES`. From this view, the character-based feature templates defined in Section 3.1 are naturally used in a word-based model.

When character-based features are incorporated into word-based CWS models, some word-based features are no longer of interest, such as the starting character of a word, sub-words contained in the word, contextual characters and so on. We consider word counting features as a complementary to character-based features, following the idea of using web-scale features in previous work, e.g. (Bansal and Klein, 2011). For a possible word w, let $count(w)$ return the count of times that w occurs as a legal word in training data. The word count number is further processed following (Bansal and Klein, 2011), $wc(w) = floor(log(count(w)) * 5)/5$. In addition to $wc(w_i)$, we also use corresponding word count features of possible words that are composed of the boundary and contextual characters of w_i. The specific word-based feature templates are illustrated in the second row of Table 5.

character--based	$\phi(c_{i_0}, \mathtt{B}, \mathbf{c}), \phi(c_{i_1}, \mathtt{M}, \mathbf{c}), \phi(c_{i_2}, \mathtt{E}, \mathbf{c})$
word--based	$wc(c_{i_0}c_{i_1}c_{i_2}), wc(c_lc_{i_0}), wc(c_{i_2}c_r)$

Table 5: Character- and word-based features of a possible word w_i over the input character sequence $\mathbf{c}$. Suppose that $w_i = c_{i_0}c_{i_1}c_{i_2}$, and its preceding and following characters are c_l and c_r respectively.

4 Training

We use the following linear model for scoring predictions: $score(\mathbf{y})=\theta^T\phi(\mathbf{x}, \mathbf{y})$, where $\phi(\mathbf{y})$ is a high-dimensional binary feature representation of $\mathbf{y}$ over input $\mathbf{x}$ and θ contains weights of these features. For parameter estimation of θ, we use the averaged perceptron as described in (Collins, 2002). This training algorithm relies on the choice of decoding algorithm. When we experiment with different decoders, by default, the parameter weights in use are trained with the corresponding decoding algorithm.

Especially, for experiments with lookahead features of English POS tagging, we prepare training data with the stacked learning technique, in order to alleviate overfitting. More specifically, we divide the training data into k folds, and tag each fold with the deterministic model learned over the other k-1 folds. The predicted tags of all folds are then merged into the gold training data and used (only) as lookahead features. Sun (2011) uses this technique to merge different levels of predictors for word segmentation.

5 Experiments

5.1 Data set

We run experiments on English POS tagging on the WSJ corpus in the Penn Treebank. Following most previous work, e.g. (Collins, 2002) and (Shen et al., 2007), we divide this corpus into training set (sections 0-18), development set (sections 19-21) and the final test set (sections 22-24).

We run experiments on Chinese word segmentation on the Penn Chinese Treebank 5.0. Following (Jiang et al., 2008a), we divide this corpus into training set (chapters 1-260), development set (chapters 271-300) and the final test set (chapters 301-325).

5.2 Deterministic constraints

Experiments in this section are carried out on the development set. The cutoff number and threshold as defined in 2.1.2, are fixed as 5 and 0.99 respectively.

	precision	recall	F_1
bigram	0.993	0.841	0.911
trigram	**0.996**	0.608	0.755
bi+trigram	0.992	**0.857**	**0.920**

Table 6: POS tagging with deterministic constraints. The maximum in each column is bold.

m_0=$\{$VBN, VBZ$\}$ & m_1=$\{$JJ, VBD, VBN$\}$ $\rightarrow$ VBN
w_0=also & m_1=$\{$VBD, VBN$\}$ $\rightarrow$ RB
m_0=$-$es & m_{-1}=$\{$IN, RB, RP$\}$ $\rightarrow$ NNS
w_0=last & w_{-1}= the $\rightarrow$ JJ

Table 7: Deterministic constraints for POS tagging.

Deterministic constraints for POS tagging

For English POS tagging, we evaluate the deterministic constraints generated by the templates described in Section 2.1.1. Since these deterministic constraints are only applied to words that occur in a constrained context, we report F-measure as the accuracy measure. Precision p is defined as the percentage of correct predictions out of all predictions, and recall r is defined as the percentage of gold predictions that are correctly predicted. F-measure F_1 is computed by $2pr/(p+r)$.

As shown in Table 6, deterministic constraints learned with both *bigram* and *trigram* templates are all very accurate in predicting POS tags of words in their context. Constraints generated by *bigram* template alone can already cover 84.1% of the input words with a high precision of 0.993. By adding the constraints generated by *trigram* template, recall is increased to 0.857 with little loss in precision. Since these deterministic constraints are applied before the decoding of probabilistic models, reliably high precision of their predictions is crucial.

There are 114589 *bigram* deterministic constraints and 130647 *trigram* constraints learned from the training data. We show a couple of examples of *bigram* deterministic constraints in Table 7. As defined in Section 2.2, we use the set of all possible POS tags for a word, e.g. $\{$VBN, VBZ$\}$, as its morph feature if the word is frequent (occurring more than 5 times in training data). For a rare word, the last two characters are used as its morph feature, e.g. $-$es. A constraint is composed of w_{-1}, w_0 and w_1, as well as the morph features m_{-1}, m_0 and m_1. For example, the first constraint in Table 7 determines the tag VBN of w_0. A deterministic constraint is aware of neither the likelihood of each possible tag or the relative rank of their likelihoods.

tagset	precision	recall	F_1
BMES	0.989	0.566	0.720
IB	**0.996**	**0.686**	**0.812**

Table 8: Character tagging with deterministic constraints.

Deterministic constraints for character tagging

For the character tagging formulation of Chinese word segmentation, we discussed two tagsets IB and BMES in Section 3.1. With respect to either tagset, we use both *bigram* and *trigram* templates to generate deterministic constraints for the corresponding tagging problem. These constraints are also evaluated by F-measure as defined above. As shown in Table 8, when tagset IB is used for character tagging, high precision predictions can be made by the deterministic constraints that are learned with respect to this tagset. However, when tagset BMES is used, the learned constraints don't always make reliable predictions, and the overall precision is not high enough to constrain a probabilistic model. Therefore, we will only use the deterministic constraints that predict IB tags in following CWS experiments.

5.3 English POS tagging

For English POS tagging, as well as the CWS problem that will be discussed in the next section, we use the development set to choose training iterations (= 5), set beam width etc. The following experiments are done on the final test set.

As introduced in Section 2.2, we adopt a very compact feature set used in (Ratnaparkhi, 1996)[1]. While searching in a constrained space, we can also extend this feature set with some basic lookahead features as defined in Section 2.2. This replicates the feature set B used in (Shen et al., 2007).

In this work, our main interest in the POS tagging problem is on its efficiency. A well-known technique to speed up Viterbi decoding is to conduct beam search. Based on experiments carried out

[1]Our implementation of this feature set is basically the same as the version used in (Collins, 2002).

Ratnaparkhi (1996)'s feature		
	Beam=1	Beam=5
raw	96.46%/3×	97.16/1×
constrained	96.80%/**14**×	**97.20**/**10**×
Feature B in (Shen et al., 2007)		
(Shen et al., 2007)	97.15% (Beam=3)	
constrained	**97.03**%/11×	**97.20**/8×

Table 9: POS tagging accuracy and speed. The maximum in each column is bold. The baseline for speed in all cases is the unconstrained tagger using (Ratnaparkhi, 1996)'s feature and conducting a beam (=5) search.

on the development set, we set beam-width of our baseline model as 5. Our baseline model, which uses Ratnaparkhi (1996)'s feature set and conducts a beam (=5) search in the unconstrained space, achieves a tagging accuracy of 97.16%. Tagging accuracy is measured by the percentage of correct predictions out of all gold predictions. We consider the speed of our baseline model as 1×, and compare other taggers with this one. The speed of a POS tagger is measured by the number of input words processed per second.

As shown in Table 9, when the beam-width is reduced from 5 to 1, the tagger (beam=1) is 3 times faster but tagging accuracy is badly hurt. In contrast, when searching in a constrained space rather than the raw space, the constrained tagger (beam=5) is 10 times fast as the baseline and the tagging accuracy is even moderately improved, increasing to 97.20%. When we evaluate the speed of a constrained tagger, the time of decoding deterministic constraints is included. These constraints make more accurate predictions than probabilistic models, thus besides improving the overall tagging speed as we expect, tagging accuracy also improves by a little.

In Viterbi decoding, all possible transitions between two neighbour states are evaluated, so the addition of locally lookahead features may have NO impact on performance. When beam-width is set to 5, tagging accuracy is not improved by the use of Feature B in (Shen et al., 2007); and because the size of the feature model grows, efficiency is hurt. On the other hand, when lookahead features are used, Viterbi-style decoding is less affected by the reduction of beam-width. As compared to the constrained greedy tagger using Ratnaparkhi (1996)'s feature set, with the additional use of three locally lookahead feature templates, tagging accuracy is increased from 96.80% to 97.02%.

When no further data is used other than training data, the bidirectional tagger described in (Shen et al., 2007) achives an accuracy of 97.33%, using a much richer feature set (E) than feature set B, the one we compare with here. As noted above, the addition of three feature templates already has a notable negative impact on efficiency, thus the use of feature set E will hurt tagging efficiency much worse. Rich feature sets are also widely used in other work that pursue state-of-art tagging accuracy, e.g. (Toutanova et al., 2003). In this work, we focus on the most compact feature sets, since tagging efficiency is our main consideration in our work on POS taging. The proposed constrained taggers as described above can achieve near state-of-art POS tagging accuracy in a much more efficient manner.

5.4 Chinese word segmentation

Like other tagging problems, Viterbi-style decoding is widely used for character tagging for CWS. We transform tagged character sequences to word segmentations first, and then evaluate word segmentations by F-measure, as defined in Section 5.2.

We proposed an ILP formulation of the CWS problem in Section 3.3, where we present a word-based model. In Section 3.4, we describe a way of mapping words to a character-based feature space. From this view, the highest scoring tagging sequence is computed subject to structural constraints, giving us an inference alternative to Viterbi decoding. For example, recall the example of input character sequence $\mathbf{c} = c_1c_2$ discussed in Section 3.3. The two possible ILP solutions give two possible segmentations $\{c_1, c_2\}$ and $\{c_1c_2\}$, thus there are 2 tag sequences evaluated by ILP, `BB` and `BI`. On the other hand, there are 4 tag sequences evaluated by Viterbi decoding: `BI`, `BB`, `IB` and `II`.

With the same feature templates as described in Section 3.1, we now compare these two decoding methods. Tagset `BMES` is used for character tagging as well as for mapping words to character-based feature space. We use the same Viterbi decoder as implemented for English POS tagging and use a non-commercial ILP solver included in GNU Linear Pro-

	precision	recall	F-measure
Viterbi	0.971	0.966	0.968
ILP	0.970	0.977	**0.974**
(Jiang et al., 2008a), POS-			0.971
(Jiang et al., 2008a), POS+			0.973

Table 10: F-measure on Chinese word segmentation. Only character-based features are used. POS-/+: perceptron trained without/with POS.

gramming Kit (GLPK), version 4.3. [2] As shown in Table 10, optimal solutions returned by an ILP solver are more accurate than optimal solutions returned by a Viterbi decoder. The F-measure is improved by a relative error reduction of 18.8%, from 0.968 to 0.974. These results are compared to the core perceptron trained without POS in (Jiang et al., 2008a). They only report results with 'lexical-target' features, a richer feature set than the one we use here. As shown in Table 10, we achieve higher performance even with more compact features.

Joint inference of CWS and Chinese POS tagging is popularly studied in recent work, e.g. (Ng and Low, 2004), (Jiang et al., 2008a), and (Kruengkrai et al., 2009). It has been shown that better performance can be achieved with joint inference, e.g. F-measure 0.978 by the cascaded model in (Jiang et al., 2008a). We focus on the task of word segmentation only in this work and show that a comparable F-measure is achievable in a much more efficient manner. Sun (2011) uses the stacked learning technique to merge different levels of predictors, obtaining a combined system that beats individual ones.

Word-based features can be easily incorporated, since the ILP formulation is more naturally viewed as a word-based model. We extend character-based features with the word count features as described in Section 3.4. Currently, we only use word counts computed from training data, i.e. still a closed test. The addition of these features makes a moderate improvement on the F-measure, from 0.974 to 0.975.

As discussed in Section 3.3, if we are able to determine that some characters always start new words, the number of possible words is reduced, i.e. the number of variables in an ILP solution is reduced. As shown in Table 11, when character sequences are partially tagged by deterministic constraints, the number of possible words per sentence, i.e. avg. $|\mathbf{x}|$, is reduced from 1290.4 to 83.7. This reduction of ILP problem size has a very important impact on the efficiency. As shown in Table 11, when taking constrained input, the segmentation speed is increased by **107** times over taking raw input, from 113 characters per second to 12,190 characters per second on a dual-core 3.0HZ CPU.

	F-measure	avg. $\|\mathbf{x}\|$	#char per sec.
raw	0.974	1290.4	113 ($1\times$)
constrained	0.974	83.75	**12190** (**107**$\times$)

Table 11: ILP problem size and segmentation speed.

Deterministic constraints predicting `IB` tags are only used here for constraining possible words. They are very accurate as shown in Section 5.2. Few gold predictions are missed from the constrained set of possible words. As shown in Table 11, F-measure is not affected by applying these constraints, while the efficiency is significantly improved.

6 Conclusion and future work

We have shown by experiments that large number of deterministic constraints can be learned from training examples, as long as the proper representation is used. These deterministic constraints are very useful in constraining probabilistic search, for example, they may be directly used for determining predictions as in English POS tagging, or used for reducing the number of variables in an ILP solution as in Chinese word segmentation. The most notable advantage in using these constraints is the increased efficiency. The two applications are both well-studied; there isn't much space for improving accuracy. Even so, we have shown that as tested with the same feature set for CWS, the proposed ILP formulation significantly improves the F-measure as compared to Viterbi decoding.

These two simple applications suggest that it is of interest to explore data-driven deterministic constraints learnt from training examples. There are more interesting ways in applying these constraints, which we are going to study in future work.

[2] `http://www.gnu.org/software/glpk`

References

M. Bansal and D. Klein. 2011. Web-scale features for full-scale parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 693–702.

Noam Chomsky. 1970. Remarks on nominalization. In R Jacobs and P Rosenbaum, editors, *Readings in English Transformational Grammar*, pages 184–221. Ginn.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, EMNLP '02, pages 1–8.

L. Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

W. Jiang, L. Huang, Q. Liu, and Y. Lü. 2008a. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *In Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

W. Jiang, H. Mi, and Q. Liu. 2008b. Word lattice reranking for chinese word segmentation and part-of-speech tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 385–392.

T. Kristjansson, A. Culotta, and P. Viola. 2004. Interactive information extraction with constrained conditional random fields. In *In AAAI*, pages 412–418.

C. Kruengkrai, K. Uchimoto, J. Kazama, Y. Wang, K. Torisawa, and H. Isahara. 2009. An error-driven word-character hybrid model for joint chinese word segmentation and pos tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, ACL '09, pages 513–521.

Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

A. F. T. Martins, N. A. Smith, and E. P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 342–350, Singapore.

H. T. Ng and J. K. Low. 2004. Chinese partof-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 277C284.

A. Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *In Proceedings of the Empirical Methods in Natural Language Processing Conference (EMNLP)*.

S. Ravi and K. Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proc. ACL*.

D. Roth and W. Yih. 2005. Integer linear programming inference for conditional random fields. In *In Proceedings of the International Conference on Machine Learning (ICML)*, pages 737–744.

L. Shen, G. Satta, and A. K. Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

R. Sproat, W. Gale, C. Shih, and N. Chang. 1996. A stochastic finite-state word-segmentation algorithm for chinese. *Comput. Linguist.*, 22(3):377–404.

W. Sun. 2011. A stacked sub-word model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of the ACL-HLT 2011*.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL-2003*.

N. Xue. 2003. Chinese word segmentation as character tagging. *International Journal of Computational Linguistics and Chinese Language Processing*, 9(1):29–48.

Y. Zhang and S. Clark. 2007. Chinese Segmentation with a Word-Based Perceptron Algorithm. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 840–847.