A Stacked Sub-Word Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging

Weiwei Sun
Department of Computational Linguistics, Saarland University
German Research Center for Artificial Intelligence (DFKI)
D-66123, Saarbrücken, Germany
wsun@coli.uni-saarland.de

Abstract

The large combined search space of joint word segmentation and Part-of-Speech (POS) tagging makes efficient decoding very hard. As a result, effective high order features representing rich contexts are inconvenient to use. In this work, we propose a novel stacked sub-word model for this task, concerning both efficiency and effectiveness. Our solution is a two step process. First, one word-based segmenter, one character-based segmenter and one local character classifier are trained to produce coarse segmentation and POS information. Second, the outputs of the three predictors are merged into sub-word sequences, which are further bracketed and labeled with POS tags by a fine-grained sub-word tagger. The coarse-to-fine search scheme is efficient, while in the sub-word tagging step rich contextual features can be approximately derived. Evaluation on the Penn Chinese Treebank shows that our model yields improvements over the best system reported in the literature.

1 Introduction

Word segmentation and part-of-speech (POS) tagging are necessary initial steps for more advanced Chinese language processing tasks, such as parsing and semantic role labeling. Joint approaches that resolve the two tasks simultaneously have received much attention in recent research. Previous work has shown that joint solutions led to accuracy improvements over pipelined systems by avoiding segmentation error propagation and exploiting POS information to help segmentation. A challenge for joint approaches is the large combined search space, which makes efficient decoding and structured learning of parameters very hard. Moreover, the representation ability of models is limited since using rich contextual word features makes the search intractable. To overcome such efficiency and effectiveness limitations, the approximate inference and reranking techniques have been explored in previous work (Zhang and Clark, 2010; Jiang et al., 2008b).

In this paper, we present an effective and efficient solution for joint Chinese word segmentation and POS tagging. Our work is motivated by several characteristics of this problem. First of all, a majority of words are easy to identify in the segmentation problem. For example, a simple maximum matching segmenter can achieve an f-score of about 90. We will show that it is possible to improve the efficiency and accuracy by using different strategies for different words. Second, segmenters designed with different views have complementary strength. We argue that the agreements and disagreements of different solvers can be used to construct an intermediate sub-word structure for joint segmentation and tagging. Since the sub-words are large enough in practice, the decoding for POS tagging over sub-words is efficient. Finally, the Chinese language is characterized by the lack of morphology that often provides important clues for POS tagging, and the POS tags contain much syntactic information, which need context information within a large window for disambiguation. For example, Huang et al. (2007) showed the effectiveness of utilizing syntactic information to rerank POS tagging results. As a result, the capability to represent rich contextual features is crucial to a POS tagger. In this work, we use a representation-efficiency tradeoff through stacked learning, a way of approximating rich *non-local* fea-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1385–1394,
Portland, Oregon, June 19-24, 2011.

tures.

This paper describes a novel stacked sub-word model. Given multiple word segmentations of one sentence, we formally define a sub-word structure that maximizes the agreement of non-word-break positions. Based on the sub-word structure, joint word segmentation and POS tagging is addressed as a two step process. In the first step, one word-based segmenter, one character-based segmenter and one local character classifier are used to produce coarse segmentation and POS information. The results of the three predictors are then merged into sub-word sequences, which are further bracketed and labeled with POS tags by a fine-grained sub-word tagger. If a string is consistently segmented as a word by the three segmenters, it will be a correct word prediction with a very high probability. In the sub-word tagging phase, the fine-grained tagger mainly considers its POS tag prediction problem. For the words that are not consistently predicted, the fine-grained tagger will also consider their bracketing problem. The coarse-to-fine scheme significantly improves the efficiency of decoding. Furthermore, in the sub-word tagging step, word features in a large window can be approximately derived from the coarse segmentation and tagging results. To train a good sub-word tagger, we use the stacked learning technique, which can effectively correct the training/test mismatch problem.

We conduct our experiments on the Penn Chinese Treebank and compare our system with the state-of-the-art systems. We present encouraging results. Our system achieves an f-score of 98.17 for the word segmentation task and an f-score of 94.02 for the whole task, resulting in relative error reductions of 14.1% and 5.5% respectively over the best system reported in the literature.

The remaining part of the paper is organized as follows. Section 2 gives a brief introduction to the problem and reviews the relevant previous research. Section 3 describes the details of our method. Section 4 presents experimental results and empirical analyses. Section 5 concludes the paper.

2 Background

2.1 Problem Definition

Given a sequence of characters $\mathbf{c} = (c_1, ..., c_{\#\mathbf{c}})$, the task of word segmentation and POS tagging is to predict a sequence of word and POS tag pairs $\mathbf{y} = (\langle w_1, p_1 \rangle, \langle w_{\#\mathbf{y}}, p_{\#\mathbf{y}} \rangle)$, where w_i is a word, p_i is its POS tag, and a "#" symbol denotes the number of elements in each variable. In order to avoid error propagation and make use of POS information for word segmentation, the two tasks should resolved jointly. Previous research has shown that the integrated methods outperformed pipelined systems (Ng and Low, 2004; Jiang et al., 2008a; Zhang and Clark, 2008).

2.2 Character-Based and Word-Based Methods

Two kinds of approaches are popular for joint word segmentation and POS tagging. The first is the "character-based" approach, where basic processing units are characters which compose words. In this kind of approach, the task is formulated as the classification of characters into POS tags with boundary information. Both the *IOB2* representation (Ramshaw and Marcus, 1995) and the *Start/End* representation (Kudo and Matsumoto, 2001) are popular. For example, the label *B-NN* indicates that a character is located at the begging of a noun. Using this method, POS information is allowed to interact with segmentation. Note that word segmentation can also be formulated as a sequential classification problem to predict whether a character is located at the beginning of, inside or at the end of a word. This character-by-character method for segmentation was first proposed in (Xue, 2003), and was then further used in POS tagging in (Ng and Low, 2004). One main disadvantage of this model is the difficulty in incorporating the whole word information.

The second kind of solution is the "word-based" method, where the basic predicting units are words themselves. This kind of solver sequentially decides whether the local sequence of characters makes up a word as well as its possible POS tag. In particular, a word-based solver reads the input sentence from left to right, predicts whether the current piece of continuous characters is a word token and which class it belongs to. Solvers may use previously predicted words and their POS information as clues to find a new word. After one word is found and classified, solvers move on and search for the next possible word. This word-by-word method for segmentation was first proposed in (Zhang and Clark, 2007),

and was then further used in POS tagging in (Zhang and Clark, 2008).

In our previous work(Sun, 2010), we presented a theoretical and empirical comparative analysis of character-based and word-based methods for Chinese word segmentation. We showed that the two methods produced different distributions of segmentation errors in a way that could be explained by theoretical properties of the two models. A system combination method that leverages the complementary strength of word-based and character-based segmentation models was also successfully explored in their work. Different from our previous focus, the diversity of different models designed with different views is utilized to construct sub-word structures in this work. We will discuss the details in the next section.

2.3 Stacked Learning

Stacked generalization is a meta-learning algorithm that was first proposed in (Wolpert, 1992) and (Breiman, 1996). The idea is to include two "levels" of predictors. The first level includes one or more predictors $g_1, ...g_K : \mathbb{R}^d \rightarrow \mathbb{R}$; each receives input $\mathbf{x} \in \mathbb{R}^d$ and outputs a prediction $g_k(\mathbf{x})$. The second level consists of a single function $h : \mathbb{R}^{d+K} \rightarrow \mathbb{R}$ that takes as input $\langle \mathbf{x}, g_1(\mathbf{x}), ..., g_K(\mathbf{x}) \rangle$ and outputs a final prediction $\hat{y} = h(\mathbf{x}, g_1(\mathbf{x}), ..., g_K(\mathbf{x}))$.

Training is done as follows. The training data $S = \{(\mathbf{x}_t, \mathbf{y}_t) : t \in [1, T]\}$ is split into L equal-sized disjoint subsets $S_1, ..., S_L$. Then functions $\mathbf{g}_1, ..., \mathbf{g}_L$ (where $\mathbf{g}_l = \langle g_1^l, ..., g_K^l \rangle$) are seperately trained on $S - S_l$, and are used to construct the augmented dataset $\hat{S} = \{(\langle \mathbf{x}_t, \hat{\mathbf{y}}_t^1, ..., \hat{\mathbf{y}}_t^K \rangle, \mathbf{y}_t) : \hat{\mathbf{y}}_t^k = g_k^l(\mathbf{x}_t)$ and $\mathbf{x}_t \in S_l\}$. Finally, each g_k is trained on the original dataset and the second level predictor h is trained on $\hat{S}$. The intent of the *cross-validation* scheme is that $\mathbf{y}_t^k$ is similar to the prediction produced by a predictor which is learned on a sample that does not include $\mathbf{x}_t$.

Stacked learning has been applied as a system ensemble method in several NLP tasks, such as named entity recognition (Wu et al., 2003) and dependency parsing (Nivre and McDonald, 2008). This framework is also explored as a solution for learning *non-local* features in (Torres Martins et al., 2008). In the machine learning research, stacked learning has been applied to structured prediction (Cohen and Carvalho, 2005). In this work, stacked learning is used to acquire extended training data for sub-word tagging.

3 Method

3.1 Architecture

In our stacked sub-word model, joint word segmentation and POS tagging is decomposed into two steps: (1) coarse-grained word segmentation and tagging, and (2) fine-grained sub-word tagging. The workflow is shown in Figure 1. In the first phase, one word-based segmenter ($\mathrm{Seg_W}$) and one character-based segmenter ($\mathrm{Seg_C}$) are trained to produce word boundaries. Additionally, a *local* character-based joint segmentation and tagging solver ($\mathrm{SegTag_L}$) is used to provide word boundaries as well as inaccurate POS information. Here, the word *local* means the labels of nearby characters are not used as features. In other words, the local character classifier assumes that the tags of characters are independent of each other. In the second phase, our system first combines the three segmentation and tagging results to get sub-words which maximize the agreement about word boundaries. Finally, a fine-grained sub-word tagger (SubTag) is applied to bracket sub-words into words and also to obtain their POS tags.

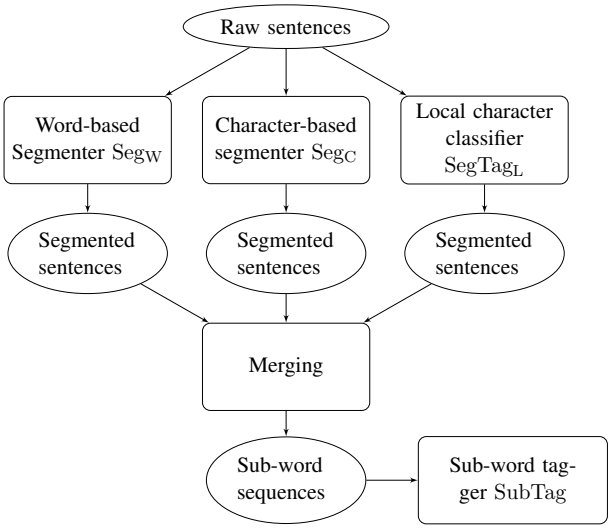

Figure 1: Workflow of the stacked sub-word model.

In our model, segmentation and POS tagging interact with each other in two processes. First, although $SegTag_L$ is locally trained, it resolves the

two sub-tasks simultaneously. Therefore, in the sub-word generating stage, segmentation and POS tagging help each other. Second, in the sub-word tagging stage, the bracketing and the classification of sub-words are jointly resolved as one sequence labeling problem.

Our experiments on the Penn Chinese Treebank will show that the word-based and character-based segmenters and the local tagger on their own produce high quality word boundaries. As a result, the oracle performance to recover words from a sub-word sequence is very high. The quality of the final tagger relies on the quality of the sub-word tagger. If a high performance sub-word tagger can be constructed, the whole task can be well resolved. The statistics will also empirically show that sub-words are significantly larger than characters and only slightly smaller than words. As a result, the search space of the sub-word tagging is significantly shrunken, and exact Viterbi decoding without approximately pruning can be efficiently processed. This property makes nearly all popular sequence labeling algorithms applicable.

Zhang et al. (2006) described a sub-word based tagging model to resolve word segmentation. To get the pieces which are larger than characters but smaller than words, they combine a character-based segmenter and a dictionary matching segmenter. Our contributions include (1) providing a formal definition of our sub-word structure that is based on multiple segmentations and (2) proposing a stacking method to acquire sub-words.

3.2 The Coarse-grained Solvers

We systematically described the implementation of two state-of-the-art Chinese word segmenters in word-based and character-based architectures, respectively (Sun, 2010). Our word-based segmenter is based on a discriminative joint model with a first order semi-Markov structure, and the other segmenter is based on a first order Markov model. Exact Viterbi-style search algorithms are used for decoding. Limited to the document length, we do not give the description of the features. We refer readers to read the above paper for details. For parameter estimation, our work adopt the *Passive-Aggressive* (PA) framework (Crammer et al., 2006), a family of margin based online learning algorithms. In this work, we introduce two simple but important refinements: (1) to shuffle the sample orders in each iteration and (2) to average the parameters in each iteration as the final parameters.

Idiom In linguistics, idioms are usually presumed to be figures of speech contradicting the principle of compositionality. As a result, it is very hard to recognize out-of-vocabulary idioms for word segmentation. However, the lexicon of idioms can be taken as a close set, which helps resolve the problem well. We collect 12992 idioms[1] from several online Chinese dictionaries. For both word-based and character-based segmentation, we first match every string of a given sentence with idioms. Every sentence is then splitted into smaller pieces which are seperated by idioms. Statistical segmentation models are then performed on these smaller character sequences.

We use a local classifier to predict the POS tag with positional information for each character. Each character can be assigned one of two possible boundary tags: "B" for a character that begins a word and "I" for a character that occurs in the middle of a word. We denote a candidate character token c_i with a fixed window $c_{i-2}c_{i-1}c_ic_{i+1}c_{i+2}$. The following features are used:

- character uni-grams: c_k $(i-2 \leq k \leq i+2)$
- character bi-grams: c_kc_{k+1} $(i-2 \leq k \leq i+1)$

To resolve the classification problem, we use the linear SVM classifier LIBLINEAR[2].

3.3 Merging Multiple Segmentation Results into Sub-Word Sequences

A majority of words are easy to identify in the segmentation problem. We favor the idea treating different words using different strategies. In this work we try to identify *simple* and *difficult* words first and to integrate them into a sub-word level. Inspired by previous work, we constructed this sub-word structure by using multiple solvers designed from different views. If a piece of continuous characters is consistently segmented by multiple segmenters, it will

[1]This resource is publicly available at http://www.coli.uni-saarland.de/~wsun/idioms.txt.

[2]Available at http://www.csie.ntu.edu.tw/~cjlin/liblinear/.

	以	总	成	绩	3	5	5	.	3	5	分	居	领	先	地	位
Answer:	[P]	[JJ]	[NN]		[			CD		]	[M]	[VV]	[JJ]		[NN]	
$\mathrm{Seg_W}$:	[]	[]	[	]	[		]	[		]	[	]	[	]	[	]
$\mathrm{Seg_C}$:	[]	[]	[	]	[						]	[]	[	]	[	]
$\mathrm{SegTag_L}$:	[P]	[JJ]	[NN]		[	CD		]	[NT]	[CD]	[NT]	[VV]	[VV]		[NN]	
Sub-words:	[P]	[JJ]	[NN]		[	B-CD	]	[I-CD]	[NT]	[CD]	[NT]	[VV]	[VV]		[NN]	

Figure 2: An example phrase: 以总成绩 3 5 5 . 3 5 分居领先地位 (Being in front with a total score of 355.35 points).

not be separated in the sub-word tagging step. The intuition is that strings which are consistently segmented by the different segmenters tend to be correct predictions. In our experiment on the Penn Chinese Treebank (Xue et al., 2005), the accuracy is 98.59% on the development data which is defined in the next section. The key point for the intermediate sub-word structures is to maximize the agreement of the three coarse-grained systems. In other words, the goal is to make merged sub-words as large as possible but not overlap with any predicted word produced by the three coarse-grained solvers. In particular, if the position between two continuous characters is predicted as a word boundary by any segmenter, this position is taken as a separation position of the sub-word sequence. This strategy makes sure that it is still possible to *re*-segment the strings of which the boundaries are disagreed with by the coarse-grained segmenters in the fine-grained tagging stage.

The formal definition is as follows. Given a sequence of characters $\mathbf{c} = (c_1, ..., c_{\#\mathbf{c}})$, let $c[i:j]$ denote a string that is made up of characters between c_i and c_j (including c_i and c_j), then a partition of the sentence can be written as $c[0:e_1], c[e_1+1:e_2], ..., c[e_m:\#\mathbf{c}]$. Let $s_k = \{c[i:j]\}$ denote the set of all segments of a partition. Given multiple partitions of a character sequence $\mathcal{S} = \{s_k\}$, there is one and only one merged partition $s_{\mathcal{S}} = \{c[i:j]\}$ s.t.

1. $\forall c[i:j] \in s_{\mathcal{S}}, \forall s_k \in \mathcal{S}, \exists c[s:e] \in s_k, s \leq i \leq j \leq e$.

2. $\forall \mathcal{C}'$ satisfies the above condition, $|\mathcal{C}'| > |\mathcal{C}|$.

The first condition makes sure that all segments in the merged partition can be only embedded in but do not overlap with any segment of any partition from $\mathcal{S}$. The second condition promises that segments of the merged partition achieve maximum length.

Figure 2 is an example to illustrate the procedure of our method. The lines $\mathrm{Seg_W}$, $\mathrm{Seg_C}$ and $\mathrm{SegTag_L}$ are the predictions of the three coarse-grained solvers. For the three words at the beginning and the two words at the end, the three predictors agree with each other. And these five words are kept as sub-words. For the character sequence " 3 5 5 . 3 5 分居", the predictions are very different. Because there are no word break predictions among the first three characters " 3 5 5 ", it is as a whole taken as one sub-word. For the other five characters, either the left position or the right position is segmented as a word break by some predictor, so the merging processor seperates them and takes each one as a single sub-word. The last line shows the merged sub-word sequence. The coarse-grained POS tags with positional information are derived from the labels provided by $\mathrm{SegTag_L}$.

3.4 The Fine-grained Sub-Word Tagger

Bracketing sub-words into words is formulated as a IOB-style sequential classification problem. Each sub-word may be assigned with one POS tag as well as two possible boundary tags: "B" for the beginning position and "I" for the middle position. A tagger is trained to classify sub-word by using the features derived from its contexts.

The sub-word level allows our system to utilize features in a large context, which is very important for POS tagging of the morphologically poor language. Features are formed making use of sub-word contents, their IOB-style inaccurate POS tags. In the following description, "C" refers to the content of the sub-word, while "T" refers to the IOB-style POS tags. For convenience, we denote a sub-word with its context $...s_{i-2}s_{i-1}s_is_{i+1}s_{i+2}...$, where s_i is

C(s_{i-1})="成绩"; T(s_{i-1})="NN" C(s_i)=" 3 5 5 "; T(s_i)="B-CD" C(s_{i+1})=". "; T(s_{i+1})="I-CD"
C(s_{i-1})C(s_i)="成绩_3 5 5 " T(s_{i-1})T(s_i)="NN_B-CD" C(s_i)C(s_{i+1})=" 3 5 5 _. " T(s_i)T(s_{i+1})="B-CD_I-CD"
C(s_{i-1})C(s_{i+1})="成绩_. " T(s_{i-1})T(s_{i+1})="B-NN_I-CD"
Prefix(1)=" 3 "; Prefix(2)=" 3 5 "; Prefix(3)=" 3 5 5 " Suffix(1)=" 5 "; Suffix(2)=" 5 5 "; Suffix(3)=" 3 5 5 "

Table 1: An example of features used in the sub-word tagging.

the current token. We denote l_C, l_T as the sizes of the window.

- Uni-gram features: C(s_k) ($-l_C \leq k \leq l_C$), T(s_k) ($-l_T \leq k \leq l_T$)
- Bi-gram features: C(s_k)C(s_{k+1}) ($-l_C \leq k \leq l_C - 1$), T(s_k)T(s_{k+1}) ($-l_T \leq k \leq l_T - 1$)
- C(s_{i-1})C(s_{i+1}) (if $l_C \geq 1$), T(s_{i-1})T(s_{i+1}) (if $l_T \geq 1$)
- T(s_{i-2})T(s_{i+1}) (if $l_T \geq 2$)
- In order to better handle unknown words, we also extract morphological features: character n-gram prefixes and suffixes for n up to 3. These features have been shown useful in previous research (Huang et al., 2007).

Take the sub-word " 3 5 5 " in Figure 2 for example, when l_C and l_T are both set to 1, all features used are listed in Table 1.

In the following experiments, we will vary window sizes l_C and l_T to find out the contribution of context information for the disambiguation. A first order Max-Margin Markov Networks model is used to resolve the sequence tagging problem. We use the SVM-HMM[3] implementation for the experiments in this work. We use the basic linear model without applying any kernel function.

[3]Available at http://www.cs.cornell.edu/People/tj/svm_light/svm_hmm.html.

Algorithm 1: The stacked learning procedure for the sub-word tagger.

input : Data $S = \{(\mathbf{c}_t, \mathbf{y}_t), t = 1, 2, ..., n\}$
Split S into L partitions $\{S_1, ...S_L\}$
for $l = 1, ..., L$ **do**
  Train $\mathrm{Seg_W}^l$, $\mathrm{Seg_C}^l$ and $\mathrm{SegTag_L}^l$ using $S - S_l$.
  Predict S_l using $\mathrm{Seg_W}^l$, $\mathrm{Seg_C}^l$ and $\mathrm{SegTag_L}^l$.
  Merge the predictions to get sub-words training sample S'_l.
end
Train the sub-word tagger SubTag using S'.

3.5 Stacked Learning for the Sub-Word Tagger

The three coarse-grained solvers $\mathrm{Seg_W}$, $\mathrm{Seg_C}$ and $\mathrm{SegTag_L}$ are directly trained on the original training data. When these three predictors are used to produce the training data, the performance is perfect. However, this does not hold when these models are applied to the test data. If we directly apply $\mathrm{Seg_W}$, $\mathrm{Seg_C}$ and $\mathrm{SegTag_L}$ to extend the training data to generate sub-word samples, the extended training data for the sub-word tagger will be very different from the data in the run time, resulting in poor performance.

One way to correct the training/test mismatch is to use the stacking method, where a K-fold *cross-validation* on the original data is performed to construct the training data for sub-word tagging. Algorithm 1 illustrates the learning procedure. First, the training data $S = \{(\mathbf{c}_t, \mathbf{y}_t)\}$ is split into L equal-sized disjoint subsets $S_1, ..., S_L$. For each subset S_l, the complementary set $S - S_l$ is used to train three coarse solvers $\mathrm{Seg_W}^l$, $\mathrm{Seg_C}^l$ and $\mathrm{SegTag_L}^l$, which process the S_l and provide inaccurate predictions. Then the inaccurate predictions are merged into sub-word sequences and S_l is extended to S'_l. Finally, the sub-word tagger is trained on the whole extended data set S'.

4 Experiments

4.1 Setting

Previous studies on joint Chinese word segmentation and POS tagging have used the Penn Chinese Treebank (CTB) in experiments. We follow this set-

ting in this paper. We use CTB 5.0 as our main corpus and define the training, development and test sets according to (Jiang et al., 2008a; Jiang et al., 2008b; Kruengkrai et al., 2009; Zhang and Clark, 2010). Table 2 shows the statistics of our experimental settings.

Data set	CTB files	# of sent.	# of words
Training	1-270	18,089	493,939
	400-931		
	1001-1151		
Devel.	301-325	350	6821
Test	271-300	348	8008

Table 2: Training, development and test data on CTB 5.0

Three metrics are used for evaluation: precision (P), recall (R) and balanced f-score (F) defined by 2PR/(P+R). Precision is the relative amount of correct words in the system output. Recall is the relative amount of correct words compared to the gold standard annotations. For segmentation, a token is considered to be correct if its boundaries match the boundaries of a word in the gold standard. For the whole task, both the boundaries and the POS tag have to be correctly identified.

4.2 Performance of the Coarse-grained Solvers

Table 3 shows the performance on the development data set of the three coarse-grained solvers. In this paper, we use 20 iterations to train $\mathrm{Seg_W}$ and $\mathrm{Seg_C}$ for all experiments. Even only locally trained, the character classifier $\mathrm{SegTag_L}$ still significantly outperforms the two state-of-the-art segmenters $\mathrm{Seg_W}$ and $\mathrm{Seg_C}$. This good performance indicates that the POS information is very important for word segmentation.

Devel.	Task	P(%)	R(%)	F
$\mathrm{Seg_W}$	Seg	94.55	94.84	94.69
$\mathrm{Seg_C}$	Seg	95.10	94.38	94.73
$\mathrm{SegTag_L}$	Seg	95.67	95.98	95.83
	Seg&Tag	87.54	91.29	89.38

Table 3: Performance of the coarse-grained solvers on the development data.

4.3 Statistics of Sub-Words

Since the base predictors to generate coarse information are two word segmenters and a local character classifier, the coarse decoding is efficient. If the length of sub-words is too short, i.e. the decoding path for sub-word sequences are too long, the decoding of the fine-grained stage is still hard. Although we cannot give a theoretical average length of sub-words, we can still show the empirical one. The average length of sub-words on the development set is 1.64, while the average length of words is 1.69. The number of all IOB-style POS tags is 59 (when using 5-fold cross-validation to generate stacked training samples). The number of all POS tags is 35. Empirically, the decoding over sub-words is $\frac{1.69}{1.64} \times (\frac{59}{35})^{n+1}$ times as slow as the decoding over words, where n is the order of the markov model. When a first order markov model is used, this number is 2.93. These statistics empirically suggest that the decoding over sub-word sequence can be efficient.

On the other hand, the sub-word sequences are not perfect in the sense that they do not promise to recover all words because of the errors made in the first step. Similarly, we can only show the empirical upper bound of the sub-word tagging. The oracle performance of the final POS tagging on the development data set is shown in Table 4. The upper bound indicates that the coarse search procedure does not lose too much.

Task	P(%)	R(%)	F
Seg&Tag	99.50%	99.09%	99.29

Table 4: Upper bound of the sub-word tagging on the development data.

One main disadvantage of character-based approach is the difficulty to incorporate word features. Since the sub-words are on average close to words, sub-word features are good approximations of word features.

4.4 Rich Contextual Features Are Useful

Table 5 shows the effect that features within different window size has on the sub-word tagging task. In this table, the symbol "C" means sub-word content features while the symbol "T" means IOB-style POS tag features. The number indicates the length

Devel.		P(%)	R(%)	F
C:±0	T:±0	92.52	92.83	92.67
C:±1	T:±0	92.63	93.27	92.95
C:±1	T:±1	92.62	93.05	92.83
C:±2	T:±0	93.17	93.86	93.51
C:±2	T:±1	93.27	93.64	93.45
C:±2	T:±2	93.08	93.61	93.34
C:±3	T:±0	93.12	93.86	93.49
C:±3	T:±1	93.34	93.96	93.65
C:±3	T:±2	93.34	93.96	93.65

Table 5: Performance of the stacked sub-word model ($K = 5$) with features in different window sizes.

of the window. For example, "C:±1" means that the tagger uses one preceding sub-word and one succeeding sub-word as features. From this table, we can clearly see the impact of features derived from neighboring sub-words. There is a significant increase between "C:±2" and "C:±1" models. This confirms our motivation that longer history and future features are crucial to the Chinese POS tagging problem. It is the main advantage of our model that making rich contextual features applicable. In all previous solutions, only features within a short history can be used due to the efficiency limitation.

The performance is further slightly improved when the window size is increased to 3. Using the labeled bracketing f-score, the evaluation shows that the "C:±3 T:±1" model performs the same as the "C:±3 T:±2" model. However, the sub-word classification accuracy of the "C:±3 T:±1" model is higher, so in the following experiments and the final results reported on the test data set, we choose this setting.

This table also suggests that the IOB-style POS information of sub-words does not contribute. We think there are two main reasons: (1) The POS information provided by the local classifier is inaccurate; (2) The structured learning of the sub-word tagger can use *real predicted* sub-word labels during its decoding time, since this learning algorithm does inference during the training time. It is still an open question whether more accurate POS information in rich contexts can help this task. If the answer is *YES*, how can we efficiently incorporate these features?

4.5 Stacked Learning Is Useful

Table 6 compares the performance of "C:±3 T:±1" models trained with no stacking as well as different folds of cross-validation. We can see that although it is still possible to improve the segmentation and POS tagging performance compared to the local character classifier, the whole task just benefits only a little from the sub-word tagging procedure if the stacking technique is not applied. The stacking technique can significantly improve the system performance, both for segmentation and POS tagging. This experiment confirms the theoretical motivation of using stacked learning: simulating the test-time setting when a sub-word tagger is applied to a new instance. There is not much difference between the 5-fold and the 10-fold cross-validation.

Devel.	Task	P(%)	R(%)	F
No stacking	Seg	95.75	96.48	96.12
	Seg&Tag	91.42	92.13	91.77
$K = 5$	Seg	96.42	97.04	96.73
	Seg&Tag	93.34	93.96	93.65
$K = 10$	Seg	96.67	97.11	96.89
	Seg&Tag	93.50	94.06	93.78

Table 6: Performance on the development data. No stacking and different folds of cross-validation are separately applied.

4.6 Final Results

Table 7 summarizes the performance of our final system on the test data and other systems reported in a majority of previous work. The final results of our system are achieved by using 10-fold cross-validation "C:±3 T:±1" models. The left most column indicates the reference of previous systems that represent state-of-the-art results. The comparison of the accuracy between our stacked sub-word system and the state-of-the-art systems in the literature indicates that our method is competitive with the best systems. Our system obtains the highest f-score performance on both segmentation and the whole task, resulting in error reductions of 14.1% and 5.5% respectively.

Test	Seg	Seg&Tag
(Jiang et al., 2008a)	97.85	93.41
(Jiang et al., 2008b)	97.74	93.37
(Kruengkrai et al., 2009)	97.87	93.67
(Zhang and Clark, 2010)	97.78	93.67
Our system	**98.17**	**94.02**

Table 7: F-score performance on the test data.

5 Conclusion and Future Work

This paper has described a stacked sub-word model for joint Chinese word segmentation and POS tagging. We defined a sub-word structure which maximizes the agreement of multiple segmentations provided by different segmenters. We showed that this sub-word structure could explore the complementary strength of different systems designed with different views. Moreover, the POS tagging could be efficiently and effectively resolved over sub-word sequences. To train a good sub-word tagger, we introduced a stacked learning procedure. Experiments showed that our approach was superior to the existing approaches reported in the literature.

Machine learning and statistical approaches encounter difficulties when the input/output data have a structured and relational form. Research in empirical Natural Language Processing has been tackling these complexities since the early work in the field. Recent work in machine learning has provided several paradigms to globally represent and process such data: linear models for structured prediction, graphical models, constrained conditional models, and reranking, among others. A general expressivity-efficiency trade off is observed. Although the stacked sub-word model is an ad hoc solution for a particular problem, namely joint word segmentation and POS tagging, the idea to employ system ensemble and stacked learning in general provides an alternative for structured problems. Multiple "cheap" coarse systems are used to provide diverse outputs, which may be inaccurate. These outputs are further merged into an intermediate representation, which allows an extractive system to use rich contexts to predict the final results. A natural avenue for future work is the extension of our method to other NLP tasks.

Acknowledgments

The work is supported by the project TAKE (Technologies for Advanced Knowledge Extraction), funded under contract 01IW08003 by the German Federal Ministry of Education and Research. The author is also funded by German Academic Exchange Service (DAAD).

The author would would like to thank Dr. Jia Xu for her helpful discussion, and Regine Bader for proofreading this paper.

References

Leo Breiman. 1996. Stacked regressions. *Mach. Learn.*, 24:49–64, July.

William W. Cohen and Vitor R. Carvalho. 2005. Stacked sequential learning. In *Proceedings of the 19th international joint conference on Artificial intelligence*, pages 671–676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *JOURNAL OF MACHINE LEARNING RESEARCH*, 7:551–585.

Zhongqiang Huang, Mary Harper, and Wen Wang. 2007. Mandarin part-of-speech tagging and discriminative reranking. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1093–1102, Prague, Czech Republic, June. Association for Computational Linguistics.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008a. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*, pages 897–904, Columbus, Ohio, June. Association for Computational Linguistics.

Wenbin Jiang, Haitao Mi, and Qun Liu. 2008b. Word lattice reranking for Chinese word segmentation and part-of-speech tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 385–392, Manchester, UK, August. Coling 2008 Organizing Committee.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi 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 Process-*

ing of the AFNLP, pages 513–521, Suntec, Singapore, August. Association for Computational Linguistics.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *NAACL '01: Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 277–284, Barcelona, Spain, July. Association for Computational Linguistics.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL-08: HLT*, pages 950–958, Columbus, Ohio, June. Association for Computational Linguistics.

L. A. Ramshaw and M. P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd ACL/SIGDAT Workshop on Very Large Corpora, Cambridge, Massachusetts, USA*, pages 82–94.

Weiwei Sun. 2010. Word-based and character-based word segmentation models: Comparison and combination. In *Coling 2010: Posters*, pages 1211–1219, Beijing, China, August. Coling 2010 Organizing Committee.

André Filipe Torres Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 157–166, Honolulu, Hawaii, October. Association for Computational Linguistics.

David H. Wolpert. 1992. Original contribution: Stacked generalization. *Neural Netw.*, 5:241–259, February.

Dekai Wu, Grace Ngai, and Marine Carpuat. 2003. A stacked, voted, stacked model for named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 200–203.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics and Chinese Language Processing*.

Yue Zhang and Stephen 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, Prague, Czech Republic, June. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896, Columbus, Ohio, June. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 843–852, Cambridge, MA, October. Association for Computational Linguistics.

Ruiqiang Zhang, Genichiro Kikui, and Eiichiro Sumita. 2006. Subword-based tagging by conditional random fields for Chinese word segmentation. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 193–196, New York City, USA, June. Association for Computational Linguistics.