Consolidation of Subtasks for Target Task in Pipelined NLP Model

Jeong-Woo Son, Heegeun Yoon, Seong-Bae Park, Keeseong Cho, and Won Ryu

Most natural language processing tasks depend on the outputs of some other tasks. Thus, they involve other tasks as subtasks. The main problem of this type of pipelined model is that the optimality of the subtasks that are trained with their own data is not guaranteed in the final target task, since the subtasks are not optimized with respect to the target task. As a solution to this problem, this paper proposes a consolidation of subtasks for a target task (CST^2). In CST^2, all parameters of a target task and its subtasks are optimized to fulfill the objective of the target task. CST^2 finds such optimized parameters through a backpropagation algorithm. In experiments in which text chunking is a target task and part-of-speech tagging is its subtask, CST^2 outperforms a traditional pipelined text chunker. The experimental results prove the effectiveness of optimizing subtasks with respect to the target task.

Keywords: Pipelined NLP model, task consolidation, chained task learning.

Manuscript received Jan. 29, 2014; revised Sept. 5, 2014; accepted Sept. 15, 2014.

This work was supported by the ICT R&D program of MSIP/IITP, Rep. of Korea (14-000-11-002, Development of Object-based Knowledge Convergence Service Platform using Image Recognition in Broadcasting Contents and 10044494, WiseKB: Big data based self-evolving knowledge base and reasoning platform).

Jeong-Woo Son (corresponding author, jwson@etri.re.kr) is with the Broadcasting & Telecommunications Media Research Laboratory, ETRI, Daejeon and also with the School of Computer Science, Kyungpook National University, Daegu, Rep. of Korea.

Heegeun Yoon (hkyoon@sejong.knu.ac.kr) and Seong-Bae Park (sbpark@sejong.knu.ac.kr) are with the College of IT Engineering, Kyungpook National University, Daegu, Rep. of Korea.

Keeseong Cho (chokis@etri.re.kr), and Won Ryu (wlyu@etri.re.kr) are with the Broadcasting & Telecommunications Media Research Laboratory, ETRI, Daejeon, Rep. of Korea.

I. Introduction

Most natural language processing (NLP) tasks consist of many pipelined subtasks. These subtasks process their own information and are often performed independently. For instance, a natural-language parser takes, in general, a tokenizer and a part-of-speech (POS) tagger as its subtasks, and these subtasks are performed sequentially and independently prior to the parser. The problem of this pipelined structure is that the errors at the subtasks are propagated to the final target task; thus, they crucially affect the performance of the task.

A general solution to this problem is to improve each subtask. Even though the subtasks are imperfect, they are optimized to perform best in their environment. However, such a subtask optimization does not guarantee the optimal performance of the target task. The objective of a subtask does not coincide with that of the target task in many cases. For instance, let us consider a POS tagger used as a subtask of text chunking. Normally, a POS tagger tries to assign a POS tag to each word in a sentence as correctly as possible. Thus, it distinguishes singular nouns, plural nouns, and proper nouns. However, the nouns need not be distinguished in text chunking, since their roles are similar in identifying a noun phrase.

Some previous studies have taken the relation among subtasks into account [1]–[3]. One way to do this is to train subtasks jointly. Florian and Ngai [4] proposed multi-dimensional transformation–based learning. They focused on the joint learning of closely related NLP tasks assuming that their order and independence are difficult to determine. On the other hand, a reuse model of subtasks has been proposed by Collobert and Weston [5]. They assumed that features that are useful for one task might also be useful for other related tasks. They therefore proposed a convolutional neural network to

http://dx.doi.org/10.4218/etrij.14.2214.0035

reuse subtasks trained for one task. However, the subtasks in these studies optimize their own objectives, not the objective of the final target task.

This paper proposes a consolidation of subtasks for a target task (CST^2). This method assumes that subtasks are cascaded to form a pipeline. That is, the output of the subtasks is used as an input of the target task. The key idea of CST^2 is that all subtasks and the target task should be optimized with respect to the objective of the target task. To achieve this goal, CST^2 optimizes the subtasks and target task iteratively. All tasks are initially trained with their own objectives. That is, the parameter values of not only the target task but also subtasks in the pipeline are determined to optimize its own objective. They are then adjusted to minimize the errors of the target task using a backpropagation algorithm. The errors at the target task are back propagated to the subtasks, and the subtasks adjust their parameters to minimize the propagated errors. This process iterates until the parameters converge.

CST^2 is evaluated using a well-known NLP task; namely, text chunking. POS tagging is used as a subtask of text chunking. The experimental results on CoNLL-2000 shared-task data show that CST^2 achieves a higher performance than the sequential model in which the output of POS tagging becomes an input of text chunking. CST^2 outperforms even a text chunker trained with true POS tags. This result proves the importance of optimization of subtasks with respect to the target task.

The remainder of this paper is organized as follows. Section II reviews other works related to joint learning in the NLP field. Section III offers an overall description of CST^2. Section IV describes how a target task and subtasks are trained in CST^2, and the experimental results are then given in Section V. Finally, some concluding remarks are provided in Section VI.

II. Related Works

NLP tasks naturally involve many other tasks as their subtasks because tasks depend on one another. That is, the output of a task often becomes an input of other tasks, and vice versa. According to [6], joint learning of tasks results in better performance than a single independent learning of each task, since the information relevant to one task may be relevant to other related tasks. Therefore, a number of techniques have been explored to learn multiple tasks simultaneously [7]–[8]. These techniques are categorized into two types: feature sharing and result sharing.

The feature-sharing type assumes that the useful features for one task are also valuable for other related tasks [4]. Therefore, the joint learning of tasks is conducted through feature sharing among tasks. Collobert and Weston [5] proposed a unified architecture for NLP tasks based on deep neural networks. In this architecture, a task generates word features at the lookup-table layer. Other tasks that are related with the task then reuse the same word features to share information among them. According to their experiments, the performance of a task is improved when it borrows the word features from another task.

On the other hand, the result-sharing type assumes that some useful features for a task are generated from the output of other related tasks. Hatori and others [9] proposed an incremental joint approach for word segmentation, POS tagging, and dependency parsing of Chinese. In this study, a unified feature set is used to learn the three tasks jointly, and the unified feature set is a composition of features from each task. As a result, each task can utilize the information generated from other tasks through a unified feature set. Goldberg and Tsarfaty [10] applied a similar idea to information sharing among morphological segmentation, POS tagging, and syntactic parsing for Hebrew. In addition, Watanabe and others [11] also proposed a similar method for learning argument role labeling and predicate sense disambiguation jointly.

Such information sharing allows substantial gain in the performance of joint tasks. However, many NLP tasks are structured as a pipelined model; that is, the target task in the pipelined model is a task that takes outputs of other subtasks as its input. The output of the target task is not fed to the subtasks. The main problem of this structure is that the errors made by the subtasks are propagated to the target task, since subtasks are usually not perfect. Khalid and others [12] reported this phenomenon in the relation of named-entity recognition (NER) and named-entity normalization (NEN). They found that most errors of NEN come from its subtask, NER. The most common way to prohibit error propagation is to optimize all subtasks as perfectly as possible [13]–[14]. However, owing to a disagreement in the objectives of the target task and its subtasks, such optimization of the subtasks does not guarantee an optimal performance of the target task.

III. Consolidation of Subtask for Target Task

Figure 1 shows the architecture of CST^2 in which one subtask is assumed. Let $S = w_1 w_2 \cdots w_N$ be an input sentence. Each word w_i in S is represented as a feature vector, $\mathbf{x}_i$. Assume that both the final target task and its subtasks are classification problems. For simplicity, we also assume that they are all binary-class classification problems.

CST^2 processes each word in the sentence sequentially. The subtask g first classifies $\mathbf{x}_i$, and the classification result is then fed forward to the target task f. Then, f determines the final classification of $\mathbf{x}_i$ using the outputs of g for $\mathbf{x}_i$ and $\mathbf{x}_{i-1}$. For

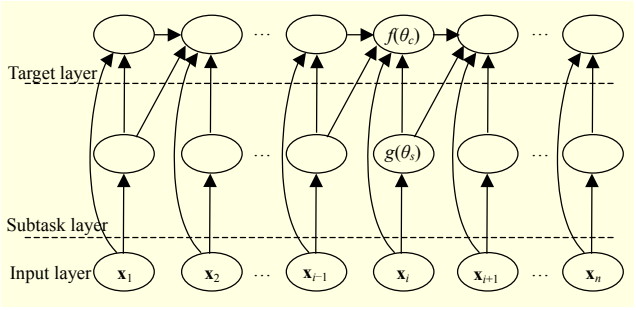


Fig. 1. Architecture of CST^2.

instance, let f be the text chunking and g be the POS tagging. A POS tagger first determines the POS tag of a given word, and the POS tags of the word and its previous word are then given to the text chunking as an input. Finally, the chunk label of the word is determined by the text chunker using the input.

If a function $f(\theta_c)$ is given to solve the target task, then the class of $\mathbf{x}_i$, y_i^*, is determined by

$$y_i^* = \underset{y\in\{+1,-1\}}{\operatorname{argmax}} \{ y = f[\theta_c; \phi(\mathbf{x}_i)] \},$$

where θ_c and $\phi(\mathbf{x}_i)$ are the parameter and feature vectors of f, respectively. The feature vector consists of the current word, $\mathbf{x}_i$, the class label of the previous word, $\mathbf{x}_{i-1}$, and the outputs of the subtask, g, for $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$. That is, when a symbol $\oplus$ represents feature concatenation, the feature vector $\phi(\mathbf{x}_i)$ is

$$\phi(\mathbf{x}_i) = \mathbf{x}_i \oplus f\left(\theta_c; \phi(\mathbf{x}_{i-1})\right) \oplus g\left(\theta_s; \mathbf{x}_{i-1}\right) \oplus g\left(\theta_s; \mathbf{x}_i\right), \quad (1)$$

where θ_s is the parameter of g. Even if f is restricted to depend on the result of one previous word in this figure, it can be generalized to depend on the previous k words without a loss of generality. In addition, this multi-layered architecture can allow more subtasks between f and g.

IV. Training CST^2

1. Parameter Optimization

The objective of CST^2 is to maximize the performance of the target task f. Thus, it optimizes both parameter θ_c of f and parameter θ_s of g by minimizing the errors of the target task f. That is, though subtask g has its own objective, it is also optimized with respect to f.

Formally, when $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_{|D|}, y_{|D|})\}$ is a training set of f, the optimal parameter of the final task f, θ_c^*, is determined by minimizing its empirical risk. That is,

$$\theta_c^* = \underset{\theta_c \in \Theta_c}{\operatorname{argmin}} \sum_i^{|D|} L(y_i, f[\theta_c; \phi(\mathbf{x}_i)]), \quad (2)$$

where Θ_c is a set of all possible θ_c. Since subtask g should be optimized with respect to f, its optimal parameter θ_s^* is also obtained by minimizing the risk of f. That is,

$$\theta_s^* = \underset{\theta_s \in \Theta_s}{\operatorname{argmin}} \sum_i^{|D|} L(y_i, f[\theta_c; \phi(\mathbf{x}_i)]), \quad (3)$$

where θ_s is a set of all possible θ_s.

The CST^2 adopts a stochastic gradient descent approach to obtain θ_c^* and θ_s^* in equations (2) and (3). Algorithm 1 shows a pseudocode for the training of CST^2. It first initializes the parameters $\theta_c^{(0)}$ and $\theta_s^{(0)}$. Both $\theta_c^{(0)}$ and $\theta_s^{(0)}$ are initialized independently to best perform their tasks f and g with their own training data. The CST^2 then updates θ_c and θ_s by adjusting them iteratively to training set D. The main loop of this algorithm iterates over the training examples in D. At time t, each input $\mathbf{x}_i$ is propagated through the architecture, and its output is predicted. In this process, to consider sequential NLP tasks, subtask g first classifies all input $\mathbf{x}_i \in D$ using $\theta_c^{(t)}$, and the target task f then predicts the class of $\mathbf{x}_i \in D$ using both $\theta_c^{(t)}$ and the results of g. After that, the error of f defined with loss L is propagated backward. The parameters $\theta_c^{(t)}$ and $\theta_s^{(t)}$ are updated in the direction of the gradient descent of the loss. This process iterates until the parameters converge.

We assume that the error of a training example $(\mathbf{x}_i, y_i)$ is defined by the square loss function. That is,

$$L(y_i, f(\theta_c; \phi(\mathbf{x}_i)) = \frac{1}{2}(f(\theta_c; \phi(\mathbf{x}_i)) - y_i)^2.$$

Algorithm 1 Training CST^2 with a stochastic gradient descent.

Input:
- $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_{|D|}, y_{|D|})\}$
- η = learning rate

Body:

Initialize $t := 0$.

Initialize parameters $\theta_c^{(0)}$ and $\theta_s^{(t)}$.

repeat

  for each $(\mathbf{x}_i, y_i) \in D$ **do**

    Propagate the input forward.

    - Execute $g(\theta_s^{(t)}; \mathbf{x}_i)$.
    - Execute $f(\theta_c^{(t)}; \phi(\mathbf{x}_i))$.

    Propagate the error backward

    - $\theta_c^{(t+1)} := \theta_c^{(t)} - \eta \cdot \nabla L(\theta_c)$.
    - $\theta_s^{(t+1)} := \theta_s^{(t)} - \eta \cdot \nabla L(\theta_s)$.

    if both θ_c and θ_s converge, **then**

      Return $(\theta_c^{(t+1)}, \theta_s^{(t+1)})$.

    end if

    $t := t + 1$.

  end for

until both θ_c and θ_s converge.

Return $(\theta_c^{(t)}, \theta_s^{(t)})$.

In addition, we restrict both f and g to a logistic function.

ETRI Journal, Volume 36, Number 5, October 2014
http://dx.doi.org/10.4218/etrij.14.2214.0035

That is,

$$f(\theta_c;\phi(\mathbf{x}_i)) = \frac{1}{1+\exp(-\theta_c^T\phi(\mathbf{x}_i))},$$

$$g(\theta_s;\mathbf{x}_i) = \frac{1}{1+\exp(-\theta_s^T\mathbf{x}_i)}.$$

Note that θ_c is a vector whose dimension is equivalent to that of $\phi(\mathbf{x}_i)$. When l is the dimension of θ_c, the gradient of $L(y_i, f[\theta_c;\phi(\mathbf{x}_i)])$ with respect to θ_c is given as

$$\nabla L(\theta_c) = \left[\frac{\partial L}{\partial\theta_c(1)}, \frac{\partial L}{\partial\theta_c(2)}, \ldots, \frac{\partial L}{\partial\theta_c(l)}\right].$$

Since f is a logistic function, the derivative $\frac{\partial L}{\partial\theta_c(j)}$ is

$$\begin{aligned}\frac{\partial L}{\partial\theta_c(j)} &= \left(f(\theta_c;\phi(\mathbf{x}_i)) - y_i\right)\cdot f(\theta_c;\phi(\mathbf{x}_i))^2 \\ &\quad\cdot\exp(-\theta_c^{\mathrm{T}}\phi(\mathbf{x}_i))\cdot\phi(\mathbf{x}_i)_j \\ &= \delta\cdot\phi(\mathbf{x}_i)_j,\end{aligned} \tag{4}$$

where $\phi(\mathbf{x}_i)_j$ is the jth feature of $\phi(\mathbf{x}_i)$ and δ is

$$\delta = \left(f(\theta_c;\phi(\mathbf{x}_i)) - y_i\right)\cdot f(\theta_c;\phi(\mathbf{x}_i))^2\cdot\exp(-\theta_c^{\mathrm{T}}\phi(\mathbf{x}_i)).$$

The parameter θ_s is also a vector whose dimension is the same as that of $\mathbf{x}_i$. It should be adjusted by the gradient descent of $L(y_i, f[\theta_c;\phi(\mathbf{x}_i)])$. Thus, $\nabla L(\theta_s)$ should be computed, but $\nabla L(\theta_s)$ cannot be obtained directly since θ_s influences f through g. We therefore use the chain rule. Then, $\frac{\partial L}{\partial\theta_s(j)}$ can be written as

$$\frac{\partial L}{\partial\theta_s(j)} = \frac{\partial L}{\partial g(\theta_s;\mathbf{x}_i)}\cdot\frac{\partial g(\theta_s;\mathbf{x}_i)}{\partial\theta_s(j)}. \tag{5}$$

The second term of this equation can be obtained in a similar way as $\frac{\partial L}{\partial\theta_c(j)}$. Therefore, we have

$$\frac{\partial g(\theta_s;\mathbf{x}_i)}{\partial\theta_s(j)} = g(\theta_s;\mathbf{x}_i)^2\cdot\exp(-\theta_s^T\mathbf{x}_i)\cdot\mathbf{x}_{ij}, \tag{6}$$

where $\mathbf{x}_{ij}$ is the jth feature value of $\mathbf{x}_i$.

The first term of (5) can be obtained using the definition of $\phi(\mathbf{x}_i,\theta_s)$ in (1). Since the derivative of L is computed with respect to $g(\theta_s;\mathbf{x}_i)$, both $f(\theta_c;\phi(\mathbf{x}_{i-1}))$ and $g(\theta_s;\mathbf{x}_{i-1})$ are regarded as a constant for all $\mathbf{x}_i$. Therefore, they are eliminated from $\frac{\partial L}{\partial g(\theta_s;\mathbf{x}_i)}$. As a result,

$$\begin{aligned}\frac{\partial L}{\partial g(\theta_s;\mathbf{x}_i)} &= \left(f(\theta_c;\phi(\mathbf{x}_i)) - y_i\right)\cdot f(\theta_c;\phi(\mathbf{x}_i))^2 \\ &\quad\cdot\exp(-\theta_c^T\phi(\mathbf{x}_i))\cdot\theta_c(r) \\ &= \delta\cdot\theta_c(r),\end{aligned} \tag{7}$$

where $\theta_c(r)$ is the rth element of θ_c that corresponds to $g(\theta_s;\mathbf{x}_i)$ in (1). From (6) and (7), $\frac{\partial L}{\partial\theta_s(j)}$ is

$$\frac{\partial L}{\partial\theta_s(j)} = g(\theta_s;\mathbf{x}_i)^2\cdot\exp(-\theta_s^T\mathbf{x}_i)\cdot\mathbf{x}_{ij}\cdot\delta\cdot\theta_c(r).$$

2. Initial Values and Multiple Classes

Since CST^2 estimates the best θ_c^* and θ_s^* by a kind of backpropagation procedure, the initial values, $\theta_c^{(0)}$ and $\theta_s^{(0)}$, can be given randomly [15]. However, decent starting values for θ_c and θ_s not only allow CST^2 to converge fast but also to converge to a better performance when many local optima exist. Therefore, we initialize θ_s by training task g sequentially with its own training data.

Assume that there is another separate training set $D' = \{(\mathbf{x}'_1, y'_1), (\mathbf{x}'_2, y'_2), \ldots, (\mathbf{x}'_{|D'|}, y'_{|D'|})\}$ for subtask g, where y'_i is the class label of $\mathbf{x}'_i$ in g. The initial value $\theta_s^{(0)}$ for parameter θ_s is then obtained by minimizing the empirical risk of g. That is,

$$\theta_s^{(0)} = \underset{\theta_s\in\Theta_s}{\operatorname{argmin}}\sum\nolimits_{(\mathbf{x}'_i,y'_i)\in D'} L(y'_i, g(\theta_s,\mathbf{x}'_i)).$$

This $\theta_s^{(0)}$ can also be obtained through a gradient descent approach. Due to this initialization step, CST^2 has another advantage; that is, models for subtasks can take benefits from its own datasets separated from the dataset for the target task.

If subtask g is a multi-class classification task, then the node

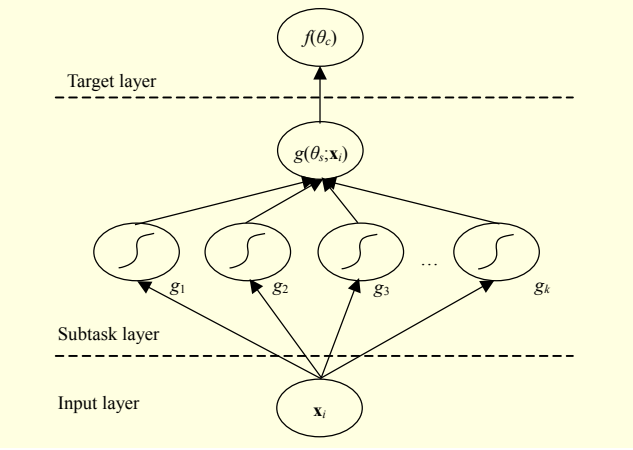


Fig. 2. CST^2 for a multi-class subtask.

for g in Fig. 1 should be changed slightly, as shown in Fig. 2. Assume that g has k classes. The subtask layer then consists of k nodes. Each $g_j (1 \le j \le k)$ in this layer is a logistic function that classifies the jth class based on a one-versus-all approach. Then g_j, whose output is largest, is determined to be $g(\theta_s; \mathbf{x}_i)$. That is,

$$g(\theta_s; x_i) = \underset{1 \le j \le k}{\operatorname{argmax}}\, g_j(\theta_s; x_i). \quad (8)$$

If the target task f is a multi-class task, then the output layer is also changed in the same manner as the multi-class subtask.

V. Characteristics of CST^2

CST^2 has mainly two characteristics compared to other learning techniques such as joint learning, multi-task learning, transfer learning, and so on. First, in CST^2, all the parameters for subtasks are actually re-estimated to support the target task through the backpropagation. Let's consider learning techniques with constraints, such as a constrained conditional model (CCM) [16]–[17]. CCM constructs subtasks independently and then combines all the tasks for the target task by using predefined constraints. Thus, the information from subtasks is selectively delivered to the target task. Since this selectivity is adopted by using constraints, CCM need not make any changes to subtasks to reflect information required for use with the target task. However, in CST^2, the information required by the target task is distributed to all subtasks by the backpropagation algorithm, and then, all parameters of subtasks are actually re-estimated.

Second, CST^2 optimizes all subtasks only for the benefit of the target task. For several tasks, CST^2 tries to achieve a maximum performance only on the target task, even though models corresponding to the subtasks sacrifice their own performance levels.

This characteristic is addressed by comparing the proposed method with multilayer sequence labeling (MSL) proposed by Azuma and Matsumoto [18]. MSL was initially designed with the same motif of CST^2 to consider pipelined tasks. However, since MSL attempts to achieve maximum performances over all tasks, its objective function still considers performances on subtasks. That is, when two tasks are given, MSL's objective function can be defined as minimizing $L_1 + L_2$, where L_1 and L_2 are loss functions for given tasks. On the other hand, CST^2 defines its objective function as minimizing L_2 if L_2 corresponds to the loss function of the target task. Due to this distinguished difference of CST^2, it is more appropriate for a system that produces the output from the target task supported by several subtasks.

VI. Experiments

1. Dataset

An evaluation of CST^2 is conducted by applying it to a well-known NLP task; text chunking.[1)] The goal of text chunking is to identify the non-recursive portions of various phrase types. Two kinds of information are, in general, used as the input of text chunking. One is lexical information, and the other is POS tagging information. Thus, POS tagging should be performed prior to text chunking and is regarded as a subtask of text chunking.

We use CoNLL-2000 shared task data as the dataset for text chunking. This dataset is constructed using a portion of the *Wall Street Journal* (WSJ) corpus. Table 2 shows simple statistics of the data sets. Sections 15–18 of the WSJ corpus are used as the training set, whereas section 20 is used as the test set. The training and test sets are composed of 211,727 words and 47,377 words, respectively. A chunk label is attached to each word using the IOB model [19]. For instance, B-NP is attached to the first word of a noun phrase, while other words in the noun phrase have I-NP as their chunk labels. Since there are eleven phrase types in the WSJ corpus, this dataset has 23 chunk labels, including chunk label O for indicating that the word is not a part of any phrase.

Most features for text chunking are borrowed from [20] and [21]. Table 1 summarizes the features used in the experiments. In this table, w_i and p_i denote the lexicon and POS tag for the ith word in a sentence, respectively. Lexicons and POS tags of word w_i and its surrounding words, w_{i+2}, are used as features.

Table 1. Features for text chunking.

Feature type	Description
Word feature	$w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, w_{i-1} \cdot w_i, w_i \cdot w_{i+1}$
Suffix	Suffixes of w_i (up to length 5)
Lexical feature	Whether w_i contains capitals Whether w_i has a number Whether w_i has a hyphen Whether w_i is all capitals Whether w_i starts with a capital and is located in the middle of a sentence
POS	$p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}, p_{i-1} \cdot p_i, p_i \cdot p_{i+1}$
Word + POS	$w_{i-1} \cdot p_i, w_i \cdot p_i, w_{i+1} \cdot p_i, p_{i-1} \cdot w_i, p_i \cdot w_i, p_{i+1} \cdot w_i$
Chunk + POS	$c_{i-1} \cdot p_i$
Chunk	$c_{i-1} \cdot c_{i-2} \cdot c_{i-1}$
Chunk + Word	$c_{i-1} \cdot w_i$

1) In all experiments, we used the same settings with [13] basically.

ETRI Journal, Volume 36, Number 5, October 2014
http://dx.doi.org/10.4218/etrij.14.2214.0035

Table 2. Simple statistics of experimental data.

		Training	Test
POS Tagging	Section	0–18	22–24
	# of sentences	38,219	5,462
	# of words	912,344	129,654
Chunking	Section	15–18	20
	# of sentences	8,936	2,012
	# of words	211,727	47,377

Table 3. Features for POS tagging.

Feature type	Description
Word feature	$w_{i-3}, w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}, w_{i+3}$, $w_{i-2} \cdot w_{i-1}, w_{i-1} \cdot w_i, w_i \cdot w_{i+1}, w_{i+1} \cdot w_{i+2}$
Prefix	Prefixes of w_i (up to length 9)
Suffix	Suffixes of w_i (up to length 9)
Lexical feature	Whether w_i contains capitals Whether w_i has a number Whether w_i has a hyphen Whether w_i is all capitals Whether w_i starts with a capital and is located in the middle of a sentence
POS	$p_{i-3}, p_{i-2}, p_{i-1}, p_{i-3} \cdot p_{i-2}, p_{i-2} \cdot p_{i-1}$
Word + POS	$p_{i-3} \cdot w_i, p_{i-2} \cdot w_i, p_{i-1} \cdot w_i$

However, only the chunk labels of previous words are used as features since text chunking works sequentially.

An independent dataset can be used for initializing θ_s of subtask g. Since POS tagging is our subtask, a standard WSJ corpus is used to train a POS tagger. The dataset is divided into training and test sets as in [22]–[23]. The training set is constructed using sections 0–18, and the test set is from sections 22–24. Table 2 shows simple statistics of the sets. The training set contains 38,219 sentences and 912,344 words, while the test set has 5,462 sentences and 129,654 words. For POS labels, 45 POS tags used in the Penn Treebank are adopted. Thus, the number of nodes in Fig. 1 is also 45. That is, $k = 45$. The features for POS tagging are those used in [24]. Table 3 explains the features. The lexicons of the current and surrounding words, as well as POS tags of previous words, are used as features. The ±3 words of the current word are used as surrounding words.

In each of the experiments, user parameter η in Algorithm 1 is heuristically set to 0.09. The performance of text chunking is measured based on the precision, recall, and F-score, whereas that of POS tagging is measured based on the accuracy.

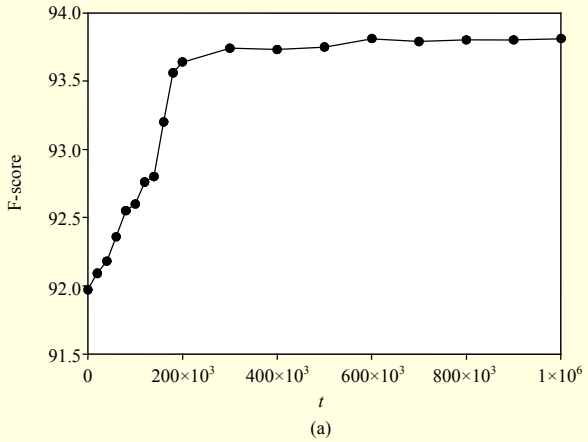


Fig. 3. CST^2 for a multi-class subtask: (a) text chunking and (b) POS tagging performances.

2. Experimental Results

Figure 3 shows the performance of CST^2. The figure shows how the performances of the target task and subtask change according to the iteration counter in Algorithm 1, (t). Figure 3(a) demonstrates that the F-score of the target task, text chunking, is improved monotonically as t increases. At $t = 0$, both text chunking and POS tagging are independently trained with their own datasets. As a result, text chunking shows a very high F-score of 91.97 in spite of its initial state. The F-score then increases up to 93.81, and becomes stabilized after $t = 6 \times 10^5$. Text chunking converges completely at $t = 10^6$. On the other hand, the performance of the subtask decreases rapidly as t increases. As shown in Fig. 3(b), the accuracy of POS tagging is 97.23% at $t = 0$, but it decreases for large t. It becomes 45.11% when $t \geq 2 \times 10^5$.

To see why the F-score of text chunking is improved whereas the accuracy of POS tagging becomes worse as t increases, we compare CST^2 with two baselines. One is CGP, which is a logistic classifier trained with gold standard POS

Table 4. Text chunking performance.

Methods	Precision	Recall	F-score
CST^2	93.15%	94.49%	93.81
CGP	92.33%	93.79%	93.05
CIP	91.23%	92.71%	91.97

Table 5. POS tags predicted by CST^2 for nouns.

Methods		Predicted tags			
		NN	NNS	NNP	NNPS
True tags	NN	6,480	0	3	0
	NNS	2,883	4	20	1
	NNP	4,141	0	361	11
	NNPS	127	0	1	2

tags, and the other is CIP, which is also a logistic classifier but trained with POS tags supplied by an independently trained POS tagger. Table 4 shows the performances of CST^2, CGP, and CIP. CIP achieves the worst F-score of 91.97, while CST^2 shows the best F-score of 93.81. Since the POS tags supplied by a trained POS tagger are used as a part of the input for CIP, some errors are included in the POS tags. These errors are the main contributing factor as to why CIP has the worst performance.

The same errors of the POS tagger are also propagated to CST^2, but because these errors are resolved by the backpropagation step, they do not have a negative effect on it. The parameter of the POS tagger is adjusted to minimize the errors of text chunking in CST^2. As a result, the F-score of CST^2 is higher than even that of CGP. CGP shows an F-score of just 93.05 even if it uses gold standard POS tags. The main reason for the lower performance of CGP than CST^2 is the over-specification of the POS tags. That is, some POS tags do not provide any information for chunk classification. For instance, there are four POS tags for nouns in the WSJ corpus: NN (singular noun), NNS (plural noun), NNP (singular proper noun), and NNPS (plural proper noun). However, there is only one NP tag for noun phrases in text chunking. All nouns become a part of NP no matter what their POS tag is. Such over-specification of POS tags, therefore, becomes redundant information for text chunking. Table 5 proves this. This table reports the predicted POS tags by CST^2 for nouns when it shows the best performance. Although true POS tags for nouns vary, CST^2 considers most of them to be NN. Therefore, the performance of POS tagging in CST^2 may be extremely low. The accuracy of the POS tagger is actually just 45.11%, as

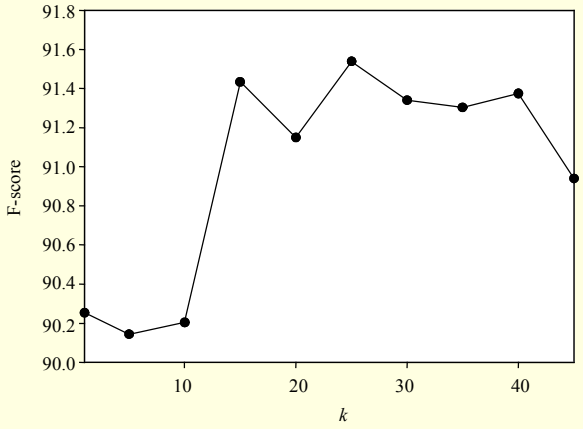


Fig. 4. Performance of random initialized CST^2 according to the number of subtask nodes.

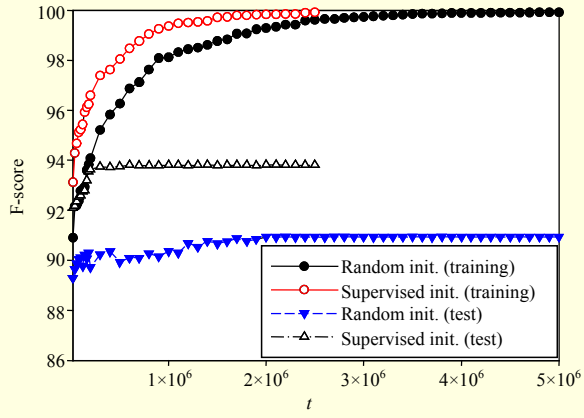


Fig. 5. Effect of supervised initialization of θ_s.

shown in Fig. 3(b). However, the fact that CST^2 outperforms both CGP and CIP proves that specific discrimination of POS tags is not needed for text chunking.

Since the number of POS tags is 45, we have 45 nodes for POS tagging. That is, $k = 45$ in Fig. 2. If some POS tags are simply redundant information, then the performance of CST^2 will be affected by the number of nodes. Figure 4 shows the F-score of CST^2 with various values of k. These F-scores are obtained after initializing both θ_c^0 and θ_s^0 randomly owing to the disagreement between k and the number of POS tags. Since different initial values can result in different local optima, the F-scores in this figure are the average of three trials. The best F-score is observed at $k = 25$. It is, therefore, proved from this figure that all POS tags are not equally significant. However, it is also revealed that even an F-score of 91.53 at $k = 25$ is still lower than that of CST^2 in Table 4. This difference is caused by different methods of parameter initialization.

Figure 5 shows how the initialization of θ_s affects CST^2.

ETRI Journal, Volume 36, Number 5, October 2014
http://dx.doi.org/10.4218/etrij.14.2214.0035

Table 6. POS tags predicted by CST^2 for nouns.

Methods	F-score
CST^2	93.81
Collobert and Weston [5]	92.71
Azuma and Matsumoto [17]	93.10
Florian and Ngai [4]	93.12
Zhang and others [25]	94.13
Kudo and Matsumoto [17]	93.91
Carreras and Márquez [26]	93.74

Supervised initialization implies that θ_s^0 is determined by training the subtask with a separate dataset. In text chunking, it is determined by a POS tagger trained using the WSJ corpus. On the other hand, a random initialization implies that θ_s^0 is chosen randomly. For both initialization methods, the number of subtask nodes is equally set to 45 ($k = 45$). In addition, the average F-score of the three trials is used for the training data.

As shown in this figure, a supervised initialization converges much faster than a random initialization. While a supervised initialization stops at around $t = 2.5 \times 10^6$, a random initialization finds a local optimum after 5×10^6 iterations. That is, a supervised initialization converges twice as fast as a random initialization. In addition, it converges to a better performance for the test data. It converges to an F-score of 93.81, whereas a random initialization stops with an F-score of just 90.94. It can therefore be induced that the training of a subtask with an independent training set is important for fast convergence to a respectable local optimum.

Finally, Table 6 compares CST^2 with legacy text chunking methods. Among the five methods compared, the works in [4], [5], and [18] utilize the joint learning of multiple NLP tasks. The remaining three are state-of-the-art methods of text chunking. According to this table, CST^2 achieves a competitive performance compared to the state-of-the-art methods, though it adopts a simple logistic function as its base classifier. In particular, even though CST^2 achieves much lower performance on POS tagging than others (accuracy of 45.11% in CST^2 versus an accuracy of 96.63% in [4], 97.19% in [5], and an F-score of 95.8 in [18]), it outperforms the three joint learning methods. These results prove the effectiveness of optimizing subtasks with respect to the objective of the target task.

VII. Conclusion

In a pipelined model of NLP tasks, the objectives of the subtasks often do not coincide with that of the target task. Thus, the subtasks optimized with their own training data do not guarantee the optimal performance of the target task. Rather, the errors of the subtasks are propagated to the target task and affect the target task negatively. To resolve this problem, we proposed a consolidation of subtasks for the target task (CST^2). The CST^2 estimates the parameters of a subtask to minimize the loss of the target task, not the loss of the subtask. For this, a back-propagation algorithm has been proposed. In this algorithm, the parameters of both the target task and subtasks are adjusted by a gradient descent of the loss function of the target task.

The CST^2 is evaluated by applying it to text chunking, in which POS tagging is a subtask. Experiments on the standard CoNLL-2000 dataset show that CST^2 can outperform both the traditional cascading method and a text chunker trained with gold standard POS tags. The F-score of text chunking in CST^2 is higher than these methods in spite of the low accuracy of POS tagging. These results prove that the POS tagger in CST^2 is specialized to text chunking. We also showed that the parameter initialization of a subtask is important for fast convergence to a respectable performance.

References

[1] G. Tur, "Multitask Learning for Spoken Language Understanding," *Proc. Int. Conf. Acoust., Speech, Signal Process.*, Toulouse, France, May 14–19, 2006, pp. 585–588.

[2] W. Sun, "A Stacked Sub-Word Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging," *Proc. Annual Meeting Association Comput. Linguistics*, Portland, OR, USA, June 19–24, 2011, pp. 1385–1394.

[3] J. Finkel and C. Manning, "Hierarchical Joint Learning: Improving Joint Parsing and Named Entity Recognition with Non-jointly Labeled Data," *Proc. Annual Meeting Association Comput. Linguistics*, Uppsala, Sweden, July 11–16, 2010, pp. 720–728.

[4] R. Florian and G. Ngai, "Multidimensional Transformation-Based Learning," *Proc. Workshop Comput. Natural Language Learning*, Boulder, CO, USA, vol. 7, June 4–5, 2009, pp. 1–8.

[5] R. Collobert and J. Weston, "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning," *Proc. Int. Conf. Mach. Learning*, Helsinki, Finland, July 5–9, 2008, pp. 160–167.

[6] R. Caruana, L. Pratt, and S. Thrun, "Multitask Learning," *Mach. Learning*, vol. 28, no. 1, July 1997, pp. 41–75.

[7] G. Neubig et al., "Unsupervised Model for Joint Phrase Alignment and Extraction," *Proc. Annual Meeting Association Comput. Linguistics*, Portland, OR, USA, vol. 1, June 19–24, 2011, pp. 632–641.

[8] X. Liu et al., "Joint Inference of Named Entity Recognition and

Normalization for Tweets," *Proc. Annual Meeting Association Comput. Linguistics*, Jeju, Rep. of Korea, vol. 1, July 8−14, 2012, pp. 526–535.

[9] J. Hatori et al., "Incremental Joint Approach to Word Segmentation, POS Tagging, and Dependency Parsing in Chinese," *Proc. Annual Meeting Association Comput. Linguistics*, Jeju, Rep. of Korea, July 8–14, 2012, pp. 1045–1053.

[10] Y. Goldberg and R. Tsarfaty, "A Single Generative Model for Joint Morphological Segmentation and Syntactic Parsing," *Proc. Annual Meeting Association Comput. Linguistics*, Columbus, OH, USA, June 15–20, 2008, pp. 371–379.

[11] Y. Watanabe, M. Asahara, and Y. Matsumoto, "A Structured Model for Joint Learning of Argument Roles and Predicate Senses," *Proc. Annual Meeting Association Comput. Linguistics*, Uppsala, Sweden, July 11–16, 2010, pp. 98–102.

[12] M. Khalid, V. Jijkoun, and M. Rijke, "The Impact of Named Entity Normalization on Information Retrieval for Question Answering," *Proc. European Conf. IR. Res*., Glasgow, UK, Mar. 30–Apr. 3, 2008, pp. 705–710.

[13] J. Finkel, C. Manning, and A. Ng, "Solving the Problem of Cascading Errors: Approximate Bayesian Inference for Linguistic Annotation Pipelines," *Proc. Conf. Empirical Methods Natural Language Process*., Sydney, Australia, July 22–23, 2006, pp. 618–626.

[14] H. Song et al., "A Cost Sensitive Part-of-Speech Tagging: Differentiating Serious Errors from Minor Errors," *Proc. Annual Meeting Association Comput. Linguistics*, Jeju, Rep. of Korea, vol. 1, July 8–14, 2012, pp. 1025–1034.

[15] S. Haykin, *Neural Networks: A Comprehensive Foundation*, New Jersey, USA: Prentice Hall, 1999.

[16] M. Chang et al., "Learning and Inference with Constraints," *Proc. AAAI Conf. Artif. Intell*., Chicago, IL, USA, July 13–17, 2008, pp. 1513–1518.

[17] D. Roth and W. Yih, "A Linear Programming Formulation for Global Inference in Natural Language Tasks," *Proc. Annual Conf. Comput. Natural Language Learning*, Boston, MA, USA, May 6–7, 2004, pp. 1–8.

[18] A. Azuma and Y. Matsumoto, "Multilayer Sequence Labelling," *Proc. Conf. Empirical Methods Natural Language Process*., Edinburgh, UK, July 30–31, 2011, pp. 628–637.

[19] L. Ramshaw and M. Marcus, "Text Chunking Using Transformation-Based Learning," *Proc. Workshop Very Large Corpora*, Boston, MA, USA, June 30, 1995, pp. 82–94.

[20] T. Kudo and Y. Matsumoto, "Chunking with Support Vector Machines," *Proc. Meeting North American Chapter Association Comput. Linguistics*, Pittsburgh, PA, USA, June 2–7, 2001, pp. 1–8.

[21] T. Kudo, *CRF++: Yet Another CRF Toolkit*, NIST, 2005. Accessed Sept. 20, 2013. http://crfpp.sourceforge.net

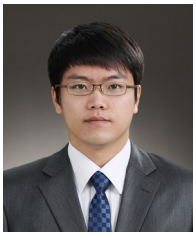
[22] K. Toutanova et al., "Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network," *Proc. Annual Meeting Association Comput. Linguistics*, Sapporo, Japan, vol. 1, July 7–12, 2003, pp. 173−180.

[23] L. Shen, G. Satta, and A. Joshi. "Guided Learning for Bidirectional Sequence Classification," *Proc. Annual Meeting Association Comput. Linguistics*, Prague, Czech Republic, June 23–30, 2007, pp. 760–767.

[24] Y. Tsuruoka and J. Tsujii, "Bidirectional Inference with the Easiest-First Strategy for Tagging Sequence Data," *Proc. Conf. Human Language Technol. Natural Language Process*., Vancouver, Canada, Oct. 6–8, 2005, pp. 467–474.

[25] T. Zhang, F. Damerau, and D. Johnson, "Text Chunking Using Regularized Winnow," *Proc. Annual Meeting Association Comput. Linguistics*, Toulouse, France, July 9–11, 2001, pp. 539–546.

[26] X. Carreras and L. Márquez, "Phrase Recognition by Filtering and Ranking with Perceptrons," *Proc. Int. Conf. Recent. Adv. Natural Language Process*., Borovets, Bulgaria, Sept. 10–12, 2003, pp. 205–216.



Jeong-Woo Son received his MS and PhD degrees in computer engineering from Kyungpook National University, Daegu, Rep. of Korea, in 2007 and 2012, respectively. Since 2013, he has been with the Broadcasting & Telecommunications Media Research Lab., Electronics and Telecommunications Research Institute, Daejeon, Rep. of Korea. He focuses on machine learning, natural language processing, and information retrieval.

Heegeun Yoon received his MS in computer engineering from Kyungpook National University, Daegu, Rep. of Korea, in 2009. Now, he is a PhD candidate at the School of Computer Science and Engineering, Kyungpook National University. His main research interests include machine learning and natural language processing.

Seong-Bae Park received his MS and PhD degrees in computer science from Seoul National University, Seoul, Rep. of Korea, in 1996 and 2006, respectively. Now, he is an associate professor at the School of Computer Science and Engineering, Kyungpook National University, Daegu, Rep. of Korea. He focuses on machine learning, natural language processing, text mining, and bio-informatics.

Keeseong Cho received his MS degree in electrical engineering from Kyungpook National University, Daegu, Rep. of Korea, in 1984. Since 1984, he has been with the Broadcasting & Telecommunications Media Research Lab., Electronics and Telecommunications Research Institute, Daejeon, Rep. of Korea. His research focuses on service control techniques; broadcasting and telecommunication convergence services; and IPTV.

Won Ryu received his MS degree in computational statistics from Seoul National University, Seoul, Rep. of Korea, in 1988 and his PhD degree in information engineering from Sungkyunkwan University, Seoul, Rep. of Korea, in 2002. Since 1989, he has been with the Intelligent Convergence Technology Research Department, Electronics and Telecommunications Research Institute, Daejeon, Rep. of Korea, where he is now a managing director. His main research interests are broadcasting and telecommunication convergence service platforms; open IPTV; and mixed reality.