Modeling Joint Entity and Relation Extraction with Table Representation

Makoto Miwa and Yutaka Sasaki
Toyota Technological Institute
2-12-1 Hisakata, Tempaku-ku, Nagoya, 468-8511, Japan
{makoto-miwa, yutaka.sasaki}@toyota-ti.ac.jp

Abstract
This paper proposes a history-based structured learning approach that jointly extracts entities and relations in a sentence. We introduce a novel simple and flexible table representation of entities and relations. We investigate several feature settings, search orders, and learning methods with inexact search on the table. The experimental results demonstrate that a joint learning approach significantly outperforms a pipeline approach by incorporating global features and by selecting appropriate learning methods and search orders.

1 Introduction
Extraction of entities and relations from texts has been traditionally treated as a pipeline of two separate subtasks: entity recognition and relation extraction. This separation makes the task easy to deal with, but it ignores underlying dependencies between and within subtasks. First, since entity recognition is not affected by relation extraction, errors in entity recognition are propagated to relation extraction. Second, relation extraction is often treated as a multi-class classification problem on pairs of entities, so dependencies between pairs are ignored. Examples of these dependencies are illustrated in Figure 1. For dependencies between subtasks, a Live_in relation requires PER and LOC entities, and vice versa. For in-subtask dependencies, the Live_in relation between “Mrs. Tsutayama” and “Japan” can be inferred from the two other relations.

Figure 1 also shows that the task has a flexible graph structure. This structure usually does not cover all the words in a sentence differently from other natural language processing (NLP) tasks such as part-of-speech (POS) tagging and dependency parsing, so local constraints are considered to be more important in the task.

Joint learning approaches (Yang and Cardie, 2013; Singh et al., 2013) incorporate these dependencies and local constraints in their models; however most approaches are time-consuming and employ complex structures consisting of multiple models. Li and Ji (2014) recently proposed a history-based structured learning approach that is simpler and more computationally efficient than other approaches. While this approach is promising, it still has a complexity in search and restricts the search order partly due to its semi-Markov representation, and thus the potential of the history-based learning is not fully investigated.

In this paper, we introduce an entity and relation table to address the difficulty in representing the task. We propose a joint extraction of entities and relations using a history-based structured learning on the table. This table representation simplifies the task into a table-filling problem, and makes the task flexible enough to incorporate several enhancements that have not been addressed in the previous history-based approach, such as search orders in decoding, global features from relations to entities, and several learning methods with inexact search.

2 Method
In this section, we first introduce an entity and relation table that is utilized to represent the whole
entity and relation structures in a sentence. We then overview our model on the table. We finally explain the decoding, learning, search order, and features in our model.

2.1 Entity and relation table

The task we address in this work is the extraction of entities and their relations from a sentence. Entities are typed and may span multiple words. Relations are typed and directed.

We use words to represent entities and relations. We assume entities do not overlap. We employ a BILOU (Begin, Inside, Last, Outside, Unit) encoding scheme that has been shown to outperform the traditional BIO scheme (Ratinov and Roth, 2009), and we will show that this scheme induces several label dependencies between words and between words and relations in §2.3.2. A label is assigned to a word according to the relative position to its corresponding entity and the type of the entity. Relations are represented with their types and directions. $\perp$ denotes a non-relation pair, and $\rightarrow$ and $\leftarrow$ denote left-to-right and right-to-left relations, respectively. Relations are defined on not entities but words, since entities are not always given when relations are extracted. Relations on entities are mapped to relations on the last words of the entities.

Based on this representation, we propose an entity and relation table that jointly represents entities and relations in a sentence. Figure 2 illustrates an entity and relation table corresponding to an example in Figure 1. We use only the lower triangular part because the table is symmetric, so the number of cells is $n(n+1)/2$ when there are $n$ words in a sentence. With this entity and relation table representation, the joint extraction problem can be mapped to a table-filling problem in that labels are assigned to cells in the table.

2.2 Model

We tackle the table-filling problem by a history-based structured learning approach that assigns labels to cells one by one. This is mostly the same as the traditional history-based model (Collins, 2002) except for the table representation.

Let $x$ be an input table, $Y(x)$ be all possible assignments to the table, and $s(x, y)$ be a scoring function that assesses the assignment of $y \in Y(x)$ to $x$. With these definitions, we define our model to predict the most probable assignment as follows:

$$y^* = \arg \max_{y \in Y(x)} s(x, y)$$

This scoring function is a decomposable function, and each decomposed function assesses the assignment of a label to a cell in the table.

$$s(x, y) = \sum_{i=1}^{\vert x \vert} s(x, y, 1, i)$$

Here, $i$ represents an index of a cell in the table, which will be explained in §2.3.1. The decomposed function $s(x, y, 1, i)$ corresponds to the $i$-th cell. The decomposed function is represented as a linear model, i.e., an inner product of features and their corresponding weights.

$$s(x, y, 1, i) = w \cdot f(x, y, 1, i)$$

The scoring function are further divided into two functions as follows:

$$s(x, y, 1, i) = s_{local}(x, y, i) + s_{global}(x, y, 1, i)$$

Here, $s_{local}(x, y, i)$ is a local scoring function that assesses the assignment to the $i$-th cell without considering other assignments, and $s_{global}(x, y, 1, i)$ is a global scoring function that assesses the assignment in the context of 1st to $(i - 1)$-th assignments. This global scoring function represents the dependencies between entities, between relations, and between entities and relations. Similarly, features $f$ are divided into local features $f_{local}$ and global features $f_{global}$, and they are defined on its target cell and surrounding contexts. The features will be explained in §2.5. The weights $w$ can also be divided, but they are tuned jointly in learning as shown in §2.4.

2.3 Decoding

The scoring function $s(x, y, 1, i)$ in Equation (2) uses all the preceding assignments and does not rely on the Markov assumption, so we cannot employ dynamic programming.

We instead employ a beam search to find the best assignment with the highest score (Collins and Roark, 2004). The beam search assigns labels to cells one by one with keeping the top $K$ best assignments when moving from a cell to the next cell, and it returns the best assignment when labels are assigned to all the cells. The pseudo code for decoding with the beam search is shown in Figure 3.
Mrs. Tsutayama is from Kumamoto Prefecture in Japan.

Mrs. B-PER
Tsutayama ⊥ L-PER
is ⊥ ⊥ O
from ⊥ ⊥ ⊥ O
Kumamoto ⊥ ... B-LOC
Prefecture ⊥ Live_in→ ⊥ ⊥ ⊥ ⊥ L-LOC
in ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ ⊥ O
Japan ⊥ Live_in→ ⊥ ⊥ ⊥ ⊥ Located_in→ ⊥ ⊥ U-LOC
.

Figure 2: The entity and relation table for the example in Figure 1.

INPUT: x: input table with no assignment, K: beam size
OUTPUT: best assignment $y^*$ for x

1: $b ← [x]$
2: for $i = 1$ to $|x|$ do
3: $T ← \emptyset$
4: for $k = 1$ to $|b|$ do
5: for $a ∈ A(i, b[k])$ do
6: $T ← T \cup \text{append}(a, b[k])$
7: end for
8: end for
9: $b ← \text{top } K \text{ tables from } T \text{ using the scoring function in Equation (2)}$
10: end for
11: return $b[0]$

Figure 3: Decoding with the beam search. $A(i, t)$ returns possible assignments for $i$-th cell of a table $t$, and append($a$, $t$) returns a table $t$ updated with an assignment $a$.

We explain how to map the table to a sequence (line 2 in Figure 3), and how to calculate possible assignments (line 6 in Figure 3) in the following subsections.

2.3.1 Table-to-sequence mapping

Cells in an input table are originally indexed in two dimensions. To apply our model in §2.2 to the cells, we need to map the two-dimensional table to a one-dimensional sequence. This is equivalent to defining a search order in the table, so we will use the terms “mapping” and “search order” interchangeably.

Since it is infeasible to try all possible mappings, we define six promising static mappings (search orders) as shown in Figure 4. Note that the “left” and “right” directions in the captions correspond to not word orders, but tables. We define two mappings (Figures 4(a) and 4(b)) with the highest priority on the “up to down” order, which checks a sentence forwardly (from the beginning of a sentence). Similarly, we also define two mappings (Figures 4(c) and 4(d)) with the highest priority on the “right to left” order, which check a sentence backwardly (from the end of a sentence). From another point of view, entities are detected before relations in Figures 4(b) and 4(c) whereas the order in a sentence is prioritized in Figures 4(a)

Figure 4: Static search orders.
Table 1: Label dependencies from relations to entities. * indicates any type.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Possible labels on $w_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation(s) on $w_{i-1}$</td>
<td>B-<em>, O, U-</em></td>
</tr>
<tr>
<td>Relation(s) on $w_i$</td>
<td>L-<em>, U-</em></td>
</tr>
</tbody>
</table>

Table 2: Label dependencies from entities to relations.

<table>
<thead>
<tr>
<th>Label on $w_i$</th>
<th>Relations from/to $w_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-<em>, I-</em>, O, L-<em>, U-</em></td>
<td>↓</td>
</tr>
<tr>
<td>L-<em>, U-</em></td>
<td>↑</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Label on $w_{i+1}$</th>
<th>Relations from/to $w_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-TYPE</td>
<td>B-<em>, O, U-</em></td>
</tr>
<tr>
<td>L-TYPE</td>
<td>L-<em>, O, U-</em></td>
</tr>
<tr>
<td>U-TYPE</td>
<td>O/S</td>
</tr>
<tr>
<td>O/S</td>
<td>B-<em>, O, U-</em></td>
</tr>
</tbody>
</table>

Table 3: Label dependencies between entities. TYPE represents an entity type, and O/S means the word is outside of a sentence.

<table>
<thead>
<tr>
<th>Label on $w_{i+2}$</th>
<th>Possible labels on $w_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-TYPE</td>
<td>B-<em>, L-</em>, L-<em>, O, U-</em></td>
</tr>
<tr>
<td>I-TYPE</td>
<td>B-, I-TYPE, L-TYPE, O, U-*</td>
</tr>
<tr>
<td>L-TYPE</td>
<td>B-, I-TYPE, L-TYPE, O, U-*</td>
</tr>
<tr>
<td>O</td>
<td>B-, L-<em>, L-</em>, O, U-*</td>
</tr>
<tr>
<td>U-TYPE</td>
<td>B-, L-<em>, L-</em>, O, U-*</td>
</tr>
<tr>
<td>O/S</td>
<td>B-, L-<em>, L-</em>, O, U-*</td>
</tr>
</tbody>
</table>

2.3.2 Label dependencies

To avoid illegal assignments to a table, we have to restrict the possible assignments to the cells according to the preceding assignments. This restriction can also reduce the computational costs.

We consider all the dependencies between cells to allow the assignments of labels to the cells in an arbitrary order. Our representation of entities and relations in §2.1 induces the dependencies between entities and between entities and relations. Tables 1-3 summarize these dependencies on the $i$-th word $w_i$ in a sentence. We can further utilize dependencies between entity types and relation types if some entity types are involved in a limited number of relation types or vice versa. We note that the dependencies between entity types and relation types include not only words participating in relations but also their surrounding words. For example, the label on $w_{i-1}$ can restrict the types of relations involving $w_i$. We employ these type dependencies in the evaluation, but we omit these dependencies here since these dependencies are dependent on the tasks.

2.4 Learning

The goal of learning is to minimize errors between predicted assignments $y^*$ and gold assignments $y^{gold}$ by tuning the weights $w$ in the scoring function in Equation 3. We employ a margin-based structured learning approach to tune the weights $w$. The pseudo code is shown in Figure 5. This approach enhances the traditional structured percep-
INPUT: training sets $D = \{(x_i, y_i)\}_{i=1}^{N}$, T: iterations
OUTPUT: weights $w$
1: $w \leftarrow 0$
2: for $t = 1$ to $T$
   3: for $x, y \in D$
   4: $y^* \leftarrow$ best assignment for $x$ using decoding in Figure 3 with $s'$ in Equation (5)
   5: if $y^* \neq y^{gold}$ then
   6: $m \leftarrow \arg \max_i \{s'(x, y^{gold}, 1, i) - s'(x, y^*, 1, i)\}$
   7: $w \leftarrow$ update($w, f(x, y^{gold}, 1, m), f(x, y^*, 1, m)$)
   8: end if
   9: end for
10: end for
11: return $w$

Figure 5: Margin-based structured learning approach with a max-violation update. update($w, f(x, y^{gold}, 1, m), f(x, y^*, 1, m)$) depends on employed learning methods.

tron (Collins, 2002) in the following ways. Firstly, we incorporate a margin $\Delta$ into the scoring function as follows so that wrong assignments with small differences from gold assignments are penalized (lines 4 and 6 in Figure 5) (Freund and Schapire, 1999).

$$s'(x, y) = s(x, y) + \Delta(y, y^{gold})$$ (5)

Similarly to the scoring function $s$, the margin $\Delta$ is defined as a decomposable function using 0-1 loss as follows:

$$\Delta(y, y^{gold}) = \sum_{i=1}^{n} \Delta(y_i, y_i^{gold}),$$

$$\Delta(y_i, y_i^{gold}) = \begin{cases} 0 & \text{if } y_i = y_i^{gold} \\ 1 & \text{otherwise} \end{cases}$$ (6)

Secondly, we update the weights $w$ based on a max-violation update rule following Huang et al. (2012) (lines 6-7 in Figure 5). Finally, we employ not only perceptron (Collins, 2002) but also AROW (Mejer and Crammer, 2010; Crammer et al., 2013), AdaGrad (Duchi et al., 2011), and DCD-SSVM (Chang and Yih, 2013) for learning methods (line 7 in Figure 5). We employ parameter averaging except for DCD-SSVM. AROW and AdaGrad store additional information for covariance and feature counts respectively, and DCD-SSVM keeps a working set and performs additional updates in each iteration. Due to space limitations, we refer to the papers for the details of the learning methods.

2.5 Features

Here, we explain the local features $f_{local}$ and the global features $f_{global}$ introduced in §2.2.

2.5.1 Local features

Our focus is not to exploit useful local features for entities and relations, so we incorporate several features from existing work to realize a reasonable baseline. Table 4 summarizes the local features. Local features for entities (or words) are similar to the features used by Florian et al. (2003), but some features are generalized and extended, and gazetteer features are excluded. For relations (or pairs of words), we employ and extend features in Miwa et al. (2009).

2.5.2 Global features

We design global features to represent dependencies among entities and relations. Table 5 summarizes the global features. These global features are activated when all the information is available during decoding. We incorporate label dependency features like traditional sequential labeling for entities. Although our model can include other non-local features between entities (Ratinov and Roth, 2009), we do not include them expecting that global features on entities and relations can cover them. We design three types of global features for relations. These features are activated when all the participating relations are not $\perp$ (non-relations). Features except for the “Crossing” category are similar to global relation features in Li and Ji (2014). We further incorporate global features for both entities and relations. These features are activated when the relation label is not $\perp$. These features can act as a bridge between entities and relations.

3 Evaluation

In this section, we first introduce the corpus and evaluation metrics that we employed for evaluation. We then show the performance on the training data set with explaining the parameters used.

---

2We tried other “Entity+Relation” features to represent a relation and both its participating entities, but they slightly degraded the performance in our preliminary experiments.
for the test set evaluation, and show the performance on the test data set.

### 3.1 Evaluation settings

We used an entity and relation recognition corpus by Roth and Yih (2004). The corpus defines four named entity types *Location*, *Organization*, *Person*, and *Other* and five relation types *Kill*, *Live In*, *Located In*, *OrgBased In* and *Work For*.

All the entities were words in the original corpus because all the spaces in entities were replaced with slashes. Previous systems (Roth and Yih, 2007; Kate and Mooney, 2010) used these word boundaries as they were, treated the boundaries as given, and focused the entity classification problem alone. Differently from such systems, we recovered these spaces by replacing these slashes with spaces to evaluate the entity boundary detection performance on this corpus. Due to this replacement and the inclusion of the boundary detection problem, our task is more challenging than the original task, and our results are not comparable with those by the previous systems.

The corpus contains 1,441 sentences that contain at least one relation. Instead of 5-fold cross validation on the entire corpus by the previous systems, we split the data set into training (1,153 sentences) and blind test (288 sentences) data sets and
developed the system on the training data set. We
tuned the hyper-parameters using a 5-fold cross
validation on the training data set, and evaluated
the performance on the test set.

We prepared a pipeline approach as a baseline. We
first trained an entity recognition model using
the local and global features, and then trained a
relation extraction model using the local features
and global features without global “Relation” fea-
tures in Table 5. We did not employ the global
“Relation” features in this baseline since it is com-
mon to treat relation extraction as a multi-class
classification problem.

We extracted features using the results from two
syntactic parsers Enju (Miyao and Tsujii, 2008)
and LRDEP (Sagae and Tsujii, 2007). We em-
ployed feature hashing (Weinberger et al., 2009)
and limited the feature space to $2^{24}$. The num-
ers of features greatly varied for categories and
targets. They also caused biased predictions that
prefer entities to relations in our preliminary ex-
periments. We thus chose to re-scale the features
as follows. We normalized local features for each
feature category and then for each target. We also
normalized global features for each feature cate-
gory, but we did not normalize them for each target
since normalization was impossible during decod-
ing. We instead scaled the global features, and the
scaling factor was tuned by using the same 5-fold
cross validation above.

We used the F1 score on relations with entities
as our primary evaluation measure and used it for
tuning parameters. In this measure, a relation with
two entities is considered correct when the offsets
and types of the entities and the type of the relation
are all correct. We also evaluated the F1 scores for
entities and relations individually on the test data
set by checking their corresponding cells. An en-
tity is correct when the offset and type are correct,
and a relation is correct when the type is correct
and the last words of two entities are correct.

3.2 Performance on Training Data Set

It is infeasible to investigate all the combinations
of the parameters, so we greedily searched for a
default parameter setting by using the evaluated
results on the training data set. The default pa-
rameter setting was the best setting except for the
beam size. We show learning curves on the train-
ing data set in Figure 6 when we varied each pa-
rameter from the default parameter setting. We
employed 5-fold cross validation. The default pa-
rameter setting used DCD-SSVM as the learning
method, entity-first, easy-first as the search order,
local and global features, and 8 as the beam size.
This section discusses how these parameters affect
the performance on the training data set and ex-
plains how the parameter setting was selected for
the test set.

Figure 6(a) compares the learning methods in-
troduced in §2.4. DCD-SSVM and AdaGrad per-
formed slightly better than perceptron, which has
often been employed in history-based structured
learning. AROW did not show comparable per-
formance to the others. We ran 100 iterations to
find the number of iterations that saturates learn-
ing curves. The large number of iterations took
time and the performance of DCD-SSVM almost
converged after 30 iterations, so we employed 50
iterations for other evaluation on the training data
set. AdaGrad got its highest performance more
quickly than other learning methods and AROW
converged slower than other methods, so we em-
ployed 10 for AdaGrad, 90 for AROW, and 50 it-
erations for other settings on the test data set.

The performance was improved by widening the
beam as in Figure 6(b), but the improvement
was gradually diminished as the beam size in-
creased. Since the wider beam requires more train-
ing and test time, we chose 8 for the beam size.

Figure 6(c) shows the effects of joint learning
as well as features explained in §2.5. We show the
performance of the pipeline approach (Pipeline)
introduced in §3.1, and the performance with lo-
cal features alone (Local), local and global fea-
tures without global “Relation” features in Table 5
(Local+global (−relation)) and all local and global
features (Local+global). We note that Pipeline
shows the learning curve of relation extraction in
the pipeline approach. Features in “Local+global
(−relation)” are the same as the features in the
pipeline approach, and the result shows that the
joint learning approach performed slightly better
than the pipeline approach. The incorporation of
global “Entity” and “Entity+Relation” features
improved the performance as is common with the
existing pipeline approaches, and relation-related
features further improved the performance.

Static search orders in §2.3.1 also affected the
performance as shown in Figure 6(d), although
search orders are not investigated in the joint en-
tity and relation extraction. Surprisingly, the gap
between the performances with the best order and worst order was about 0.04 in an F1 score, which is statistically significant, and the performance can be worse than the pipeline approach in Figure 6(c). This means improvement by joint learning can be easily cancelled out if we do not carefully consider search order. It is also surprising that the second worst order (Figure 4(b)) is the most intuitive “left-to-right” order, which is closest to the order in Li and Ji (2014) among the six search orders.

Figure 6(e) shows the performance with dynamic search orders. Unfortunately, the easy-first policy did not work well on this entity and relation task, but, with the two enhancements, dynamic orders performed as well as the best static order in Figure 6(d). This shows that entities should be de-
3.3 Performance on Test Data Set

Table 6 summarizes the performance on the test data set. We employed the default parameter setting explained in §3.2, and compared parameters by changing the parameters shown in the first column. We performed a statistical test using the approximate randomization method (Noreen, 1989) on our primary measure (“Entity+Relation”). The results are almost consistent with the results on the training data set with a few exceptions.

Differently from the results on the training data set, AdaGrad and AROW performed significantly worse than perceptron and DCD-SSVM and they performed slightly worse than the pipeline approach. This result shows that DCD-SSVM performs well with inexact search and the selection of learning methods can significantly affect the entity and relation extraction performance.

The joint learning approach showed a significant improvement over the pipeline approach with relation-related global features, although the joint learning approach alone did not show a significant improvement over the pipeline approach. Unfortunately, no joint learning approach outperformed the pipeline approach in entity recognition. This may be partly because hyper-parameters were tuned to the primary measure. The results on the pipeline approach also indicate that the better performance on entity recognition does not necessarily improve the relation extraction performance.

Search orders also affected the performance, and the worst order (right to left, down to up) and best order (close-first, left to right) were significantly different. The performance of the worst order was worse than that of the pipeline approach, although the difference was not significant. These results show that it is necessary to carefully select the search order for the joint entity and relation extraction task.

3.4 Comparison with Other Systems

To compare our model with the other systems (Roth and Yih, 2007; Kate and Mooney, 2010), we evaluated the performance of our model when the entity boundaries were given. Differently from our setting in §3.1, we used the gold entity boundaries encoded in the BILOU scheme and assigned entity labels to the boundaries. We performed 5-fold cross validation on the data set following Roth and Yih (2007) although the split was different from theirs since their splits were not available. We employed the default parameter setting in §3.2 for this comparison.

Table 7 shows the evaluation results. Although we cannot directly compare the results, our model performs better than the other models. Compared to Table 6, Table 7 also shows that the inclusion of entity boundary detection degrades the performance about 0.09 in F-score.

4 Related Work

Search order in structured learning has been studied in several NLP tasks. Left-to-right and right-to-left orderings have been often investigated in sequential labeling tasks (Kudo and Matsumoto, 2001). Easy-first policy was firstly introduced by Goldberg and Elhadad (2010) for dependency parsing, and it was successfully employed in several tasks, such as joint POS tagging and dependency parsing (Ma et al., 2012) and co-reference resolution (Stoyanov and Eisner, 2012). Search order, however, has not been focused in relation extraction tasks.

Named entity recognition (Florian et al., 2003; Nadeau and Sekine, 2007) and relation extraction (Zelenko et al., 2003; Miwa et al., 2009) have often been treated as separate tasks, but there are some previous studies that treat entities and relations jointly in learning. Most studies built joint learning models upon individual models for subtasks, such as Integer Linear Programming (ILP) (Roth and Yih, 2007; Yang and Cardie, 2013) and Card-Pyramid Parsing (Kate and Mooney, 2010). Our approach does not require such individual models, and it also can detect entity boundaries that these approaches except for Yang and Cardie (2013) did not treat. Other studies (Yu and Lam, 2010; Singh et al., 2013) built global probabilistic graphical models. They need to compute distributions over variables, but our approach does not. Li and Ji (2014) proposed an approach to jointly find entities and relations. They incorporated a semi-Markov chain in representing entities and they defined two actions during search, but our approach does not employ such representation and actions, and thus it is more simple and flexible to investigate search orders.

5 Conclusions

In this paper, we proposed a history-based structured learning approach that jointly detects enti-
of entities and relations. We introduced a novel entity and relation table that jointly represents entities and relations, and showed how the entity and relation extraction task can be mapped to a simple table-filling problem. We also investigated search orders and learning methods that have been fixed in previous research. Experimental results showed that the joint learning approach outperforms the pipeline approach and the appropriate selection of learning methods and search orders is crucial to produce a high performance on this task.

As future work, we plan to apply this approach to other relation extraction tasks and explore more suitable search orders for relation extraction tasks.

We also plan to investigate the potential of this table representation in other tasks such as semantic parsing and co-reference resolution.

Acknowledgments

We thank Yoshimasa Tsuruoka and Yusuke Miyao for valuable discussions, and the anonymous reviewers for their insightful comments. This work was supported by the TTI Start-Up Research Support Program and the JSPS Grant-in-Aid for Young Scientists (B) [grant number 25730129].

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Entity</th>
<th>Relation</th>
<th>Entity+Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron</td>
<td>0.809 / 0.809 / 0.809</td>
<td>0.760 / 0.547 / 0.636</td>
<td>0.731 / 0.527 / 0.612</td>
</tr>
<tr>
<td>AdaGrad</td>
<td>0.801 / 0.790 / 0.795</td>
<td>0.732 / 0.486 / 0.584</td>
<td>0.716 / 0.476 / 0.572</td>
</tr>
<tr>
<td>AROW</td>
<td>0.810 / 0.802 / 0.806</td>
<td>0.797 / 0.468 / 0.590</td>
<td>0.758 / 0.445 / 0.561</td>
</tr>
<tr>
<td>DCD-SSVM†</td>
<td>0.810 / 0.802 / 0.807</td>
<td>0.783 / 0.524 / 0.628</td>
<td>0.760 / 0.509 / 0.610*</td>
</tr>
<tr>
<td>Pipeline</td>
<td>0.823 / 0.814 / 0.818</td>
<td>0.672 / 0.542 / 0.600</td>
<td>0.647 / 0.522 / 0.577</td>
</tr>
<tr>
<td>Local</td>
<td>0.819 / 0.812 / 0.815</td>
<td>0.844 / 0.399 / 0.542</td>
<td>0.812 / 0.384 / 0.522</td>
</tr>
<tr>
<td>Local + global (−relation)</td>
<td>0.809 / 0.799 / 0.804</td>
<td>0.784 / 0.481 / 0.596</td>
<td>0.747 / 0.458 / 0.568</td>
</tr>
<tr>
<td>Local + global †</td>
<td>0.812 / 0.802 / 0.807</td>
<td>0.783 / 0.524 / 0.628</td>
<td>0.760 / 0.509 / 0.610*</td>
</tr>
<tr>
<td>(a) Up to down, left to right</td>
<td>0.824 / 0.801 / 0.813</td>
<td>0.821 / 0.433 / 0.567</td>
<td>0.787 / 0.415 / 0.543</td>
</tr>
<tr>
<td>(b) Up to down, right to left</td>
<td>0.828 / 0.808 / 0.818</td>
<td>0.850 / 0.461 / 0.597</td>
<td>0.822 / 0.445 / 0.578</td>
</tr>
<tr>
<td>(c) Right to left, up to down</td>
<td>0.823 / 0.799 / 0.811</td>
<td>0.826 / 0.448 / 0.581</td>
<td>0.789 / 0.427 / 0.554</td>
</tr>
<tr>
<td>(d) Right to left, down to up</td>
<td>0.811 / 0.784 / 0.797</td>
<td>0.774 / 0.445 / 0.565</td>
<td>0.739 / 0.425 / 0.540</td>
</tr>
<tr>
<td>(e) Close-first, left to right</td>
<td>0.821 / 0.806 / 0.813</td>
<td>0.807 / 0.522 / 0.634</td>
<td>0.780 / 0.504 / 0.612*</td>
</tr>
<tr>
<td>(f) Close-first, right to left</td>
<td>0.817 / 0.801 / 0.809</td>
<td>0.832 / 0.491 / 0.618</td>
<td>0.797 / 0.471 / 0.592</td>
</tr>
<tr>
<td>Easy-first</td>
<td>0.811 / 0.790 / 0.801</td>
<td>0.862 / 0.415 / 0.560</td>
<td>0.831 / 0.399 / 0.540</td>
</tr>
<tr>
<td>Entity-first, easy-first †</td>
<td>0.812 / 0.802 / 0.807</td>
<td>0.783 / 0.524 / 0.628</td>
<td>0.760 / 0.509 / 0.610*</td>
</tr>
<tr>
<td>Close-first, easy-first</td>
<td>0.816 / 0.803 / 0.810</td>
<td>0.796 / 0.486 / 0.603</td>
<td>0.767 / 0.468 / 0.581</td>
</tr>
</tbody>
</table>

Table 6: Performance of entity and relation extraction on the test data set (precision / recall / F1 score). The † denotes the default parameter setting in §3.2 and * represents a significant improvement over the underlined “Pipeline” baseline (p<0.05). Labels (a)-(f) correspond to those in Figure 4.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Person</td>
<td>0.921 / 0.942 / 0.932</td>
<td>0.891 / 0.895 / 0.890</td>
<td>0.931 / 0.948 / 0.939</td>
</tr>
<tr>
<td>Location</td>
<td>0.908 / 0.942 / 0.924</td>
<td>0.897 / 0.887 / 0.891</td>
<td>0.922 / 0.939 / 0.930</td>
</tr>
<tr>
<td>Organization</td>
<td>0.905 / 0.887 / 0.895</td>
<td>0.895 / 0.720 / 0.792</td>
<td>0.903 / 0.896 / 0.899</td>
</tr>
<tr>
<td>All entities</td>
<td>-</td>
<td>-</td>
<td>0.924 / 0.924 / 0.924</td>
</tr>
<tr>
<td>Located_In</td>
<td>0.675 / 0.567 / 0.583</td>
<td>0.539 / 0.557 / 0.513</td>
<td>0.821 / 0.549 / 0.654</td>
</tr>
<tr>
<td>Work_For</td>
<td>0.735 / 0.683 / 0.707</td>
<td>0.720 / 0.423 / 0.531</td>
<td>0.886 / 0.642 / 0.743</td>
</tr>
<tr>
<td>OrgBased_In</td>
<td>0.662 / 0.641 / 0.647</td>
<td>0.798 / 0.416 / 0.543</td>
<td>0.768 / 0.572 / 0.654</td>
</tr>
<tr>
<td>Live_In</td>
<td>0.664 / 0.601 / 0.629</td>
<td>0.591 / 0.490 / 0.530</td>
<td>0.819 / 0.532 / 0.644</td>
</tr>
<tr>
<td>Kill</td>
<td>0.916 / 0.641 / 0.752</td>
<td>0.775 / 0.815 / 0.790</td>
<td>0.933 / 0.797 / 0.858</td>
</tr>
<tr>
<td>All relations</td>
<td>-</td>
<td>-</td>
<td>0.837 / 0.599 / 0.698</td>
</tr>
</tbody>
</table>

Table 7: Results of entity classification and relation extraction on the data set using the 5-fold cross validation (precision / recall / F1 score).
References


