Semi-supervised Noun Compound Analysis with Edge and Span Features

Yugo MURAWAKI   Sadao KUROHASHI
Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
{murawaki, kuro}@i.kyoto-u.ac.jp

Abstract

In this paper, we propose the use of *spans* in addition to edges in noun compound analysis. A span is a sequence of words that can represent a noun compound. Compared with edges, spans have good properties in terms of semi-supervised parsing. They can be reliably extracted from a huge amount of unannotated text. In addition, while the combinations of edges such as sibling and grandparent interactions are, in general, difficult to handle in parsing, it is quite easy to utilize spans with arbitrary width. We show that spans can be incorporated straightforwardly into the standard chart-based parsing algorithm. We create a semi-supervised discriminative parser that combines edge and span features. Experiments show that span features improve accuracy and that further gain is obtained when they are combined with edge features.

Title and Abstract in Japanese

スパンとエッジ特徴量を用いた半教師あり名詞句解析

名詞句解析において，エッジだけでなくスパンを手がかりとして使うことを提案する．スパンは名詞句を表しうる単語列であり，エッジと比べて半教師あり学習に適した性質を持っている．すなわち，大量の生テキストから高い信頼性をもって抽出可能である．さらに，エッジは解析時に組み合わせ (兄弟や孫の関係など) を考えることが一般に難しいのに対して，スパンは任意の長さの組み合わせを自明に利用できる．この論文では，スパンが動的計画法による標準的な構文解析手法に簡単に組み込めることを示し，エッジとスパン特徴量を組み合わせた半教師ありの識別型構文解析器を提案する．実験により，スパン特徴量が解析精度を改善し，エッジと組み合わせることでさらに精度が向上することが示された．

Keywords: noun compound analysis, semi-supervised learning, parsing, span.

Keywords in Japanese: 名詞句解析, 半教師あり学習, 構文解析, スパン.

1 Introduction

Words are used as a basic unit in a broad range of applications in natural language processing. However, it often happens that what we need to recognize turn out to be longer than single words. They are phrases, noun compounds in particular, that consist of more than one word.

A noun compound is not just a sequence of words but has a latent structure. Consider the following example in Japanese.

[*jidou* [*onsei* *ninshiki*]]
automation speech recognition
automatic speech recognition

The brackets indicate the internal structure of the noun compound. In addition to the right-branching structure, it has another possible interpretation.

[[*jidou* *onsei*] *ninshiki*]
automation speech recognition
recognition of automatic speech

Our goal is to recognize that the former is semantically coherent while the latter is not. In order to analyze the internal structures of noun compounds, we need some automatic method because they are too large in number and too productive to be covered by a hand-crafted lexicon. This task is called noun compound analysis.

Noun compound analysis can be seen as a task of dependency parsing (Lauer, 1995). However, it is different from usual full-sentence parsing in that part-of-speech tags help little, if at all. A noun compound is just a sequence of nouns and lacks grammatical markers. For this reason we take fully lexicalized approaches in noun compound analysis.

Fully lexicalized approaches often suffer from the data sparseness problem. Apart from the observation that nouns have much higher domain specificity than other words, a model needs to learn the lexical association of a pair of words. However, we can never create an annotated corpus that covers the combinations of tens of thousands of words.

To overcome data sparseness, it seems promising to take semi-supervised approaches that exploit a huge amount of unannotated text. In fact, recent studies have shown that web statistics greatly improve the accuracy of noun compound analysis (Lapata and Keller, 2004; Nakov and Hearst, 2005a; Bergsma et al., 2010; Pitler et al., 2010).

One problem with incorporating web statistics into the dependency model is that dependency relations (edges) are latent and cannot be observed in unannotated text. Edge counts are approximated by bigrams of successive words (Nakov and Hearst, 2005a) or rely on a search engine's `NEAR` operator (Lapata and Keller, 2004). These counts are noisy as they may not represent true dependency relations.

In this paper, we propose the use of *spans* in addition to edges. A span is a sequence of words that can form a noun compound. Spans have two advantages over edges. First, unlike noisy edges, spans can be reliably extracted from unannotated text without abandoning a large portion of data. Second, it is quite easy to handle spans with arbitrary width in a parsing model. We show that web span counts can be used straightforwardly in the standard chart-based parsing algorithm. By contrast, combinations of edges such as sibling and grandparent interactions cannot easily be incorporated into dynamic programming.

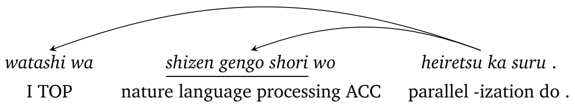


Figure 1: *Bunsetsu*-based dependency parsing for “I parallelize natural language processing.” The dependency relations are drawn between *bunsetsu* phrases. The internal structure of the noun compound (underlined) is left unanalyzed.

We create a semi-supervised discriminative parser that can combine multiple factors: features learned directly from training data, web-derived edge features and web-derived span features. In addition, we introduce web-derived paraphrase features, which can be seen as mixtures of edges and spans.

Experiments show that span features improve accuracy and are robust across domains. It is also shown that the edge and span features play complementary roles. The combination of these features boost performance in out-of-domain data.

2 Related work

2.1 Background of the task

The phrase structure grammars dominated English parsing research for a long time although dependency parsing has seen rapid progress in the last decade. The most influential annotated corpus for the phrase structure grammars would be the Penn Treebank (Marcus et al., 1993). Unfortunately, the original Penn Treebank does not annotate the internal structures of noun compounds but leaves them flat. The situation changed when Vadas and Curran (2007a) added internal structures to noun compounds in the Penn Treebank and gave rise to supervised approaches to English noun compound analysis (Vadas and Curran, 2007b; Bergsma et al., 2010; Pitler et al., 2010).

Japanese parsing faces a similar situation but dependency parsing is the preferred choice in order to handle its flexible word order (Uchimoto et al., 1999; Kudo and Matsumoto, 2002). In Japanese dependency parsing, dependency relations are drawn between phrasal units called *bunsetsu* although there is an attempt at word (morpheme)-based dependency parsing (Flannery et al., 2011). In a *bunsetsu* phrase, one or more content words are followed by zero or more function words (morphemes). This means that, as illustrated in Figure 1, the internal structure of a noun compound is left unanalyzed because it is contained in a *bunsetsu* phrase. Thus noun compound analysis has a complementary relationship with full-sentence dependency parsing.

2.2 Noun compound analysis

Noun compound analysis, also called noun compound bracketing, is the task of analyzing the internal structure of a given noun compound. There is an old debate between what are called the adjacency model and the dependency model. Figure 2 compares the two models for three-word noun compounds, which were the primary focus of early work. The adjacency model (Marcus, 1980; Liberman and Sproat, 1992; Pustejovsky et al., 1993; Resnik, 1993) examines the lexical association between neighboring words. It checks if the pair of N_2 and N_3 is more strongly associated than the pair of N_1 and N_2. The dependency model (Lauer, 1995) compares the association between N_1 and N_3 against that between N_1 and N_2. Lauer (1995) and subsequent studies demonstrate that the dependency model performs better than

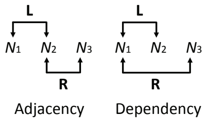


Figure 2: Adjacency and dependency models. A reproduction of Figure 1 in Lauer (1995).

the adjacency model. In this paper, however, we show that a generalization of adjacency is useful for noun compound analysis.

Since dependency relations (edges) are latent, annotated data are required to learn the true lexical association. Otherwise we need some approximation methods with which the lexical association is estimated from unannotated text. In an unsupervised setting, Lauer (1995) investigated two methods of approximation:

1. counts from two-word noun compounds, and
2. co-occurrences of a pair of nouns within some fixed window.

The former was shown to outperform the latter.

Subsequent studies have focused on the use of web statistics. We can today obtain huge amounts of text from the web. With this situation, various studies in various fields of natural language processing report performance improvement with the use of web-scale text (Banko and Brill, 2001; Brants et al., 2007; Sasano et al., 2009). In noun compound analysis, Lapata and Keller (2004) and Nakov and Hearst (2005a) used search engine hit counts. Bergsma et al. (2010) and Pitler et al. (2010) utilized Google's N-gram.

Web-based approaches need approximation methods because the web is essentially unannotated text. Lapata and Keller (2004) used a search engine's `NEAR` operator, which might correspond to co-occurrence statistics. Nakov and Hearst (2005a) relied on phrase search. The result can be interpreted as bigrams of successive words. Bergsma et al. (2010) and Pitler et al. (2010) seem to have used bigram counts (plus unigram counts to calculate probabilities).

2.3 Use of arbitrarily sized chunks

The use of arbitrarily sized chunks is relatively new in natural language analysis. Traditionally it has been done by decomposing an input into minimal elements. In probabilistic context-free grammars, for example, the probability of generating a tree is the product of the probabilities of generating each derivation rule. Similarly, a first-order dependency parser defines the score of a dependency tree as the sum of the score of all edges in the tree (McDonald et al., 2005).

The use of arbitrarily sized chunks resulted in a huge success in statistical machine translation (Koehn et al., 2003). Recent studies successfully make use of arbitrarily sized chunks in various tasks of natural language analysis too. Chunks range from sequences (Wood et al., 2011) to tree fragments (Post and Gildea, 2009) and subtrees (Johnson et al., 2007) of phrase-structure grammars. They are usually realized by non-parametric Bayesian models, which provide a way to balance between data fitting and model complexity.

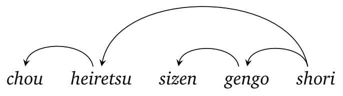
A drawback of non-parametric Bayesian inference is high computational cost that makes it difficult to scale to the web. This is especially the case when Markov chain Monte Carlo sampling is used for inference because it is difficult to parallelize in a theoretically sound way. For this reason, we seek a different kind of statistics that are applicable to a huge amount of web text.

chou *heiretsu* *sizen* *gengo* *shori*
super parallel nature language processing
massively parallel natural language processing

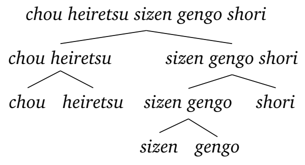
(a)

[[*chou heiretsu*] [[*sizen gengo*] *shori*]]

(b)



(c)



(d)

Figure 3: Various representations of a noun compound. (a) Word sequence with word-by-word and full translations. It is taken as the input by our parser. (b) Bracketed representation for its internal structure. (c) Equivalent dependency tree with edges. (d) Span-based binary tree representation.

3 Noun compound analysis

3.1 Task settings

In noun compound analysis, the model takes each noun compound as input and outputs its internal structure. The input is a word sequence as shown in Figure 3a. Since Japanese does not delimit words by white-space, we assume that a word sequence is provided either manually or by some automatic analyzer.

The internal structure of a noun compound can be denoted by brackets (Figure 3b). Bracketing of a noun compound can equivalently be represented as a binary tree. We assume the head-final order for dependency. In other words, a non-final word always modifies a word on its right. With this assumption, we can transform the bracketing structure of a noun compound into an equivalent dependency tree (Figure 3c) and thus noun compound bracketing can be formalized as a parsing problem. Also we can avoid some complex issues that arise from bidirectional parsing (Eisner and Satta, 1999; Johnson, 2007).[1] Alternatively we can use a span-based binary tree representation (Figure 3d). As the output we may think of any representation above.

Formally, our parser is given a word sequence $\boldsymbol{n} = n_1, \cdots, n_L$ as input. Its goal is to output the correct tree $\boldsymbol{t}$. We ignore noun compounds if length $L < 3$ because we assume that their structures are unambiguous. We consider a (semi-)supervised setting. The annotated noun compounds $\mathscr{T} = \{(\boldsymbol{n}_i, \boldsymbol{t}_i)\}_{i=1}^{T}$ are used to train our parser.

3.2 Initial dependency parser

We treat noun compound analysis as a structure prediction problem. Previous studies focused on three word noun compounds that only require a single binary decision per input (Lauer, 1995; Lapata and Keller, 2004; Nakov and Hearst, 2005a; Bergsma et al., 2010), handled longer noun compounds but relied on a series of local decisions (Barker, 1998; Vadas and Curran, 2007b), or used a pseudo-generative model based on a local discriminative classifier (Pitler et al., 2010). By contrast, we directly score entire trees for a given noun compound. We do not

[1]We speculate that our span features, described below, are technically applicable to bidirectional parsing. We leave the application to non-Japanese languages for future work.

Base
$\langle d \rangle$
$\langle n_j, d \rangle$
$\langle n_k, d \rangle$
$\langle n_j, n_k \rangle$
$\langle n_j, n_k, d \rangle$

(a)

TWNC
: $\mathit{log1p}(c_{\mathrm{TWNC}}(n_j, n_k))$
LTW
: $\mathit{log1p}(c_{\mathrm{LTW}}(n_j, n_k))$
χ^2 bigram
: $\mathit{log1p}(\chi^2(n_j, n_k))$
PMI cooc
: $\mathrm{PMI}(n_j, n_k)$
PMI_UNK

(b)

Span
$\langle {}_iS_k \rangle$

Web span
$\langle s \rangle$: $\mathit{log1p}(c_{\mathrm{SPAN}}({}_iS_k))$

(c)

Paraphrase
: $\mathit{log1p}(c_{\mathrm{PARA}}({}_iS_j, {}_{j+1}S_k))$
: $\mathit{log1p}(c_{\mathrm{PARA}}(n_j, {}_{j+1}S_k))$
: $\mathit{log1p}(c_{\mathrm{PARA}}({}_iS_j, n_k))$
: $\mathit{log1p}(c_{\mathrm{PARA}}(n_j, n_k))$

(d)

Table 1: Features used by our parser. We consider the combination of spans ${}_iS_j$ and ${}_{j+1}S_k$, and an edge is drawn between n_j and n_k. $\langle x \rangle$ denotes a template that is expanded into multiple features. The left-hand side of the colon is the feature's name. Omitted when obvious. The right-hand side is the feature's value. Binary-valued when omitted. (a) Base edge features. $d = k - j$ is the distance between n_j and n_k (1, 2, 3, 4 or ≥ 5). (b) Web-derived edge features. (c) Span features. $s = k - i + 1$ is the width of the span ${}_iS_k$ (2, 3, 4, 5 or ≥ 6). (d) Paraphrase features.

re-impelemt earlier models but, for comparison, incorporate them as features of our parser.

We begin with a first-order projective dependency model (Eisner, 1996; McDonald et al., 2005), which will be extended later. Specifically we use a high-dimensional linear classifier. The score of a dependency tree is defined as the sum of the score of all edges in the tree,

$$score(\boldsymbol{n}, \boldsymbol{t}) = \sum_{(j,k) \in edges(\boldsymbol{t})} \boldsymbol{w} \cdot \boldsymbol{\phi}(j,k)$$

where $edges(\boldsymbol{t})$ returns all edges in $\boldsymbol{t}$, $\boldsymbol{\phi}(j,k)$ gives a feature vector for the edge between n_j and n_k, and $\boldsymbol{w}$ is the corresponding weight vector that will be learned during training.

Table 1a shows features used by the initial parser, all of which are binary-valued. Unlike full-sentence parsing, noun compound analysis heavily relies on the edge distance d because an overwhelming majority of non-final words modify words to their immediate right. Also d helps the model capture some suffix-like words' tendency to being modified by their left-hand neighbors.

Given $\boldsymbol{w}$ and $\boldsymbol{n}$, we want to find $\boldsymbol{t}$ such that

$$\boldsymbol{t} = \operatorname*{argmax}_{\boldsymbol{t}'} score(\boldsymbol{n}, \boldsymbol{t}').$$

Following (McDonald et al., 2005), we adopt a lexicalized CKY chart parsing algorithm. Just like the one for context-free grammars, our algorithm uses bottom-up dynamic programming. For the word sequence $n_1, \cdots, n_L$, we consider a *span* ${}_iS_j = n_i, \cdots, n_j$ $(i \leq j)$, which holds a score. Our algorithm is simpler than that of McDonald et al. (2005) because we assume the head-final order. Thus whereas bidirectional parsing needs to keep 3 indices for each span, we only need one.

We begin with single-word spans, iteratively combine a pair of spans ${}_iS_j$ and ${}_{j+1}S_k$ to create a larger span ${}_iS_k$, and end up with ${}_1S_L$, the span for the whole word sequence. When ${}_iS_j$ and ${}_{j+1}S_k$ are combined, an edge is drawn between n_j and n_k. Given j, the score of ${}_iS_k$ is the sum of its own edge score and the scores of ${}_iS_j$ and ${}_{j+1}S_k$. To create ${}_iS_k$, we check every possible pair of subspans ${}_iS_j$ and ${}_{j+1}S_k$ by iterating over j and select the one best.

Algorithm 1 Passive-aggressive training (PA-I, prediction-based updates, weight averaging).

```
Input: training data $\mathscr{T} = \{(\boldsymbol{n}_i, \boldsymbol{t}_i)\}_{i=1}^{T}$
 1: $\boldsymbol{w} = \boldsymbol{0}$; $\boldsymbol{v} = \boldsymbol{0}$
 2: for $n = 1..N$ do
 3:     shuffle $\mathscr{T}$
 4:     for $(\boldsymbol{n}, \boldsymbol{t}) \in \mathscr{T}$ do
 5:         predict $\hat{\boldsymbol{t}} = \text{argmax}_t\ score(\boldsymbol{n}, \boldsymbol{t})$
 6:         calculate cost $\rho$, the number of misidentified edges
 7:         if $\rho > 0$ then
 8:             loss $l = score(\boldsymbol{n}, \hat{\boldsymbol{t}}) - score(\boldsymbol{n}, \boldsymbol{t}) + \rho$
 9:             $\tau = \min\{C, l/\|\Phi(\boldsymbol{n}, \boldsymbol{t}) - \Phi(\boldsymbol{n}, \hat{\boldsymbol{t}})\|^2\}$
10:             $\boldsymbol{w} = \boldsymbol{w} + \tau(\Phi(\boldsymbol{n}, \boldsymbol{t}) - \Phi(\boldsymbol{n}, \hat{\boldsymbol{t}}))$
11:             $\boldsymbol{v} = \boldsymbol{v} + \boldsymbol{w}$
12:         end if
13:     end for
14: end for
15: $\boldsymbol{w} = \boldsymbol{v}/(N * T)$
```

We use an online learning algorithm for training. We implement the online passive-aggressive algorithm (Crammer et al., 2006). Specifically we use the variant named PA-I, with prediction-based updates (Crammer et al., 2006) and weight averaging (Collins, 2002). Algorithm 1 gives the pseudo-code, in which $\Phi(\boldsymbol{n}, \boldsymbol{t}) = \sum_{(j,k)\in edges(\boldsymbol{t})} \boldsymbol{\phi}(j, k)$. We set C as 1.0.

3.3 Noun compound extraction

Next we extend the initial parser with web statistics. While previous studies relied on search engine hit counts (Lapata and Keller, 2004; Nakov and Hearst, 2005a) or n-grams (Bergsma et al., 2010; Pitler et al., 2010), we directly utilize an unannotated text corpus.

In preparation for calculating web statistics, we extract noun compounds from the web corpus. To do this, we first apply the morphological analyzer JUMAN[2] to each sentence to segment it into a word sequence. We then use the dependency parser KNP[3] to identify noun compounds. At a pre-processing step before dependency parsing, KNP chunks a given word sequence into *bunsetsu* phrases. We examine each nominal *bunsetsu* phrase, drop function words that follow content words, and extract sequences of noun and noun-like words. Extracted noun compounds are clean since word segmentation and phrase chunking can be done highly accurately. Note that extracted noun compounds include two-word ones.

3.4 Web-derived edge features

3.4.1 Two-word noun compounds (TWNC and LTW)

Using extracted noun compounds, we introduce web-derived edge features that measure the association between child n_j and head n_k. Following Lauer (1995), we begin with the simplest method of approximation, namely the use of counts from two-word noun compounds. Let $c_{\text{TWNC}}(n_j, n_k)$ be the number of two-word noun compounds that consist of n_j and n_k. We take the log of $c_{\text{TWNC}}(n_j, n_k)$ (to be precise, we use $log1p(x) = log(1 + x)$ to avoid zeros) and

[2]`http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN`
[3]`http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?KNP`

add the result as one additional feature of the parser (**TWNC**, first of Table 1b). As usual, its corresponding weight is tuned using training data.

One advantage of this method is that the result is very clean because we only use a reliable portion of data. However, it has more chance of suffering from the problem of data sparseness than methods that exploit full data. Certain dependency relations might appear only in noun compounds with three or more words.

Alternatively, we focus on the last two words of every noun compound, which in our assumption always have a dependency relation. We take *log1p* of $c_{\text{LTW}}(n_j, n_k)$, the number of such word pairs (**LTW**, second of Table 1b).

3.4.2 Chi-squared bigram measure (χ^2 bigram)

Nakov and Hearst (2005a) used the bigram count $c(n_j, n_k)$, or the number of pages returned by a search engine in response to queries for the exact phrase "n_j n_k." Instead we collect all bigrams from noun compounds extracted from the web corpus. Either way, bigrams are not clean because they may not represent true dependency relations.

Nakov and Hearst (2005a) empirically showed that the χ^2 dependency measure performed better than other measures. The χ^2 measure is defined as follows:

$$\chi^2(n_j, n_k) = \frac{N(AD - BC)^2}{(A + C)(B + D)(A + B)(C + D)}$$

where $A = c(n_j, n_k)$, $B = c(n_j, \overline{n_k})$, $C = c(\overline{n_j}, n_k)$, $D = c(\overline{n_j}, \overline{n_k})$, and $N = A + B + C + D$ ($c(n_j, \overline{n_k})$ is the number of bigrams in which n_j is followed by a word other than n_k). Zero counts are replaced by 0.5. We take *log1p* of the χ^2 measure and add the result as one additional feature of the parser (χ^2 **bigram**, third of Table 1b).

3.4.3 PMI co-occurrence measure (PMI cooc)

Co-occurrences of n_j and n_k within noun compounds are yet another option although co-occurrences are also rough approximations of dependency relations. Since co-occurrence statistics require much larger space than successive bigrams, we only store pairs of words whose co-occurrence counts are greater than or equal to 10.

We follow Pitler et al. (2010) and use the pointwise mutual information (PMI)[4]

$$\text{PMI}(n_j, n_k) = log \frac{p_{\text{cooc}}(n_j, n_k)}{p_{\text{left}}(n_j) p_{\text{right}}(n_k)},$$

where $p_{\text{cooc}}(n_j, n_k)$ is the probability of the pair n_j, n_k appearing in the same noun compound in this order, $p_{\text{left}}(n_j)$ is the probability of n_j appearing in the left side of co-occurrence pairs, and $p_{\text{right}}(n_k)$ is defined in a similar manner.

We append PMI(n_j, n_k) as a new feature. Another feature PMI_UNK is fired alternatively if $p_{\text{cooc}}(n_j, n_k) = 0$, $p_{\text{left}}(n_j) = 0$ or $p_{\text{right}}(n_k) = 0$ (**PMI cooc**, last of Table 1b).

[4]We could try any combination of (1) χ^2 and PMI, and (2) bigrams and co-occurrences. We did not investigate this further because experiments showed that simple log-counts performed very well.

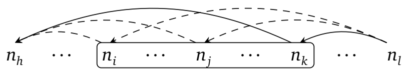


Figure 4: Span as constraints. The box denotes span ${}_iS_k$. The dashed edges are ruled out by the span while the solid ones are still possible.

3.5 Span features

The crux of our parsing model is the use of span features in addition to edge features. To do this, we first rewrite the score function in accordance with the parsing algorithm:

$$score(\boldsymbol{n}, \boldsymbol{t}) = \sum_{(i,j,k) \in spans(\boldsymbol{t})} \boldsymbol{w} \cdot \boldsymbol{\phi}(i, j, k)$$

where $spans(\boldsymbol{t})$ returns all spans with width ≥ 2, which we call non-trivial spans. Non-trivial spans correspond to non-leaf nodes in Figure 3d. The 3-tuple (i, j, k) represents each non-trivial span ${}_iS_k$ that consists of two subspans ${}_iS_j$ and ${}_{j+1}S_k$ $(i \leq j \leq k)$. Edge features introduced in Sections 3.2 and 3.4 only use j (child) and k (head).

Now we introduce span features, which add a score to ${}_iS_k$. It is clear that even with this extension, we can still use the CKY algorithm for parsing. For each non-trivial span ${}_iS_k$, we add its span score only after selecting the best pair of subspans ${}_iS_j$ and ${}_{j+1}S_k$ by iterating over j because span features do not depend on j. The training algorithm is the same as before.

The selection of a span constrains multiple edges at once as illustrated in Figure 4. For span ${}_iS_k$, only n_k can interact with the outside of the span: n_k can be modified by n_h $(h < i)$, and n_k can modify n_l $(k < l)$. However, n_j $(i \leq j < k)$ cannot be modified by n_h or cannot modify n_l. This property is useful when, for example, the widely used term "*jidou onsei ninshiki*" (automatic speech recognition) is followed by "*shisutemu*" (system). The selection of a span for the first three words rules out the edge between "*jidou*" (automatic) and "*shisutemu*" (system), and the edge between "*onsei*" (speech) and "*shisutemu*" (system) although both edges are plausible when context is ignored.

We employ two types of span features (**Span** and **Web span** in Table 1c). One is a set of binary features, each of which corresponds to a non-trivial span appearing in the training data. The other is a set of web-derived features, grouped by span width. We take *log1p* of $c_{\text{SPAN}}({}_iS_k)$, the number of times ${}_iS_k$ appears as the noun compound in the web corpus. For $c_{\text{SPAN}}({}_iS_k)$, we do not consider a noun compound nested in a longer noun compound because they cannot be identified confidently. Thus web-derived spans are clean even though we exploit the whole data.

Span features can be seen as a generalization of adjacency employed in early studies (Lauer, 1995). While adjacency is the measure of association between two successive words, span features cover not only two successive words but longer word sequences. As seen above, the parser can easily handle spans with arbitrary width. By contrast, combinations of edges can be handled with dynamic programming only if they are restricted to certain patterns such as consecutive siblings and grandparents (McDonald and Pereira, 2006; Koo and Collins, 2010).

3.6 Paraphrase features

In preliminary experiments, we discovered that many noun compounds took the form of predicate-argument pair. Such a predicate noun compound typically contains a *sahen* noun, which functions as a verb when followed by light verbs such as *suru* (to do), *dekiru* (can do) and *sareru* (to be done). A nominal predicate-argument pair can be paraphrased by the combination of a noun phrase and a verbal phrase. For example, the noun compound "*sizen gengo shori*" (natural language processing) has corresponding explanatory expressions including

sizen	*gengo*	*wo*	‖ *shori*	*suru*
nature	language	ACC	processing	do

to process natural language(s)

(‖ denotes a phrasal boundary). Conversely, this expression suggests the bracket structure "[[*sizen gengo*] *shori*]" ([[natural language] processing]).

We collect pairs of predicate and argument noun compounds from the web corpus and incorporate them as paraphrase features. We follow Kawahara and Kurohashi (2001) for extracting predicate-argument pairs. Although parsing errors are inevitable, we can circumvent this problem by exploiting the constraints of Japanese dependency structures: head-final and projective. The simplest example would be the second-to-last phrase of a sentence, which always depends on the last phrase. With such constraints, we can focus on syntactically unambiguous dependency pairs. For a newspaper corpus, 20.7% of dependency relations are extracted and their accuracy is 98.3% (Kawahara, 2012, p.c.).

For an argument noun phrase, we accept it only if it has the nominative (*ga*) or accusative (*wo*) case marker. For a predicate, we do not distinguish the type of light verbs ("to do," "can do", "to be done" and others). Note that predicate noun compounds may be longer than one word. The following examples are two-word noun compounds that can be used as predicates.

soshiki	*ka*
organ	-ization

organization (the act of organizing)

shouryou	*seisan*
little-volume	production

little-volume production

Paraphrase features conform with the three-argument feature function $\boldsymbol{\phi}(i,j,k)$ and can be used straightforwardly in dynamic programming. We employ four features as shown in Table 1d. The first feature takes *log1p* of $c_{\text{PARA}}({}_iS_j, {}_{j+1}S_k)$, the number of times span ${}_iS_j$ is used as an argument of a verbal phrase derived from span ${}_{j+1}S_k$. The second feature is based on $c_{\text{PARA}}(n_j, {}_{j+1}S_k)$, which resembles that of the first feature but uses the head word n_j instead of the span ${}_iS_j$. The third and fourth features are defined in similar manners. These features are mixtures of edges and spans.

Nakov and Hearst (2005a) incorporated hyphen, concatenation and other paraphrase-based cues into noun compound bracketing. For example, *cell-cycle* and *healthcare* reinforce the bracket structures "[[*cell cycle*] *analysis*]" and "[[*health care*] *reform*]" respectively. They have no Japanese counterpart, however. Other tasks of linguistic analysis in which simple paraphrase features are used include word segmentation (Kaji and Kitsuregawa, 2011) and PP attachment (Nakov and Hearst, 2005b; Bansal and Klein, 2011).

4 Experiments

4.1 Data

In-domain data We first built annotated data for both training and testing. We used the NTCIR1 TMREC test collection (Kageura et al., 1999).[5] It consisted of 1,870 Japanese paper

[5] We chose this test collection simply because it can also be used in future research on applications of noun compound analysis.

abstracts in the field of computer science. Gold-standard segmentation and POS tagging were provided.[6]

Following Nakagawa and Mori (2002), we extracted uninterrupted noun sequences as noun compounds. We discarded chunking errors, and single-word and two-word noun compounds. We randomly selected 3,100 noun compounds and manually annotated them with dependency relations.

Out-of-domain data We constructed two sets of annotated noun compounds for testing. We used J-STAGE,[7] an online collection of electronic journals. We collected Japanese papers in the fields of agriculture (**Out-of-domain 1**) and material science (**Out-of-domain 2**).

We extracted noun compounds through the procedure described in Section 3.3. We discarded segmentation and chunking errors, and two-word noun compounds. We randomly selected 1,000 noun compounds for each set and manually annotated them with dependency relations.

Web corpus The web corpus from which web statistics was calculated was compiled through procedures proposed by Kawahara and Kurohashi (2006). It consisted of about 70 million Japanese web pages.

4.2 Models

We trained and tested the parser with various combinations of features. For each model, we run 20 iterations for online learning. We conducted 5-fold cross-validation on the in-domain data. For out-of-domain data, we trained each model on the whole in-domain data.

For comparison, we also examined three baseline methods.
Left-branching Every non-final word modifies its immediate right neighbor.
Right-branching Every non-final word modifies the final word.
Random Choose a dependency tree at random.

4.3 Evaluation measures

We measured the performance of the parser with unlabeled attachment score (UAS). UAS is defined as the proportion of correctly identified dependency relations. Note that we did not exclude the second last word, which in our assumption always modified the last word. We allowed annotators to break the assumption although we found none. We used McNemar's test of significance to evaluate the degree of difference between a pair of model outputs.

4.4 Results

Table 2 shows unlabeled attachment scores. The left-branching baseline was strong because an overwhelming majority of non-final words modified their immediate right neighbors. The discriminative parser managed to beat the left-branching baseline even with the **Base** features alone.

Not surprisingly, porting to out-of-domain data resulted in drops in accuracy. These disparities can be explained by the fact that while for in-domain data, 64.6% of edges (word pairs, regardless of distance) in test data were observed at least once in training data, the number dropped drastically to 5.0% and 5.4% for out-of-domain data.

[6]Segmentations were sometimes inconsistent with those of the morphological analyzer JUMAN, which was used for building web statistics. This might have a slightly unfavorable impact on performance.

[7]`https://www.jstage.jst.go.jp/browse/`

Model	In-domain (7,717 edges)	Out-of-domain 1 (2,370 edges)	Out-of-domain 2 (2,389 edges)
Left-branching	88.32	86.20	86.40
Right-branching	49.03	53.54	52.66
Random	68.47	69.11	69.61
Base	94.27	88.44	88.32
+ TWNC	94.32	90.17**	92.42**
+ LTW	94.27	88.31	89.74**
+ χ^2 bigram	94.13	87.43	88.70
+ PMI cooc	94.30	87.93	88.91
+ Span	94.40	88.23	88.87*
+ Web span	94.35	88.86	90.41**
+ Span + Web span	94.43	88.86	90.83**
+ Paraphrase	94.08	88.06	88.74
+ Span + Web span + TWNC	94.54	90.13**	92.21**
+ Span + Web span + Paraphrase	94.46	89.49*	91.53**
+ Span + Web span + TWNC + Paraphrase	**94.64***	**90.30****	**93.01****

Table 2: Unlabeled attachment scores. * and ** mark statistically significant improvement over the **Base** model with $p < 0.05$ and $p < 0.01$ respectively.

Among the four types of web-derived edge features, the simplest **TWNC** feature performed best, consistently improving accuracy. The gains obtained for out-of-domain data were remarkable. Somewhat unexpectedly, the **LTW** feature performed much worse than **TWNC**. The χ^2 **bigram** and **PMI cooc** features were consistently beaten by **TWNC**.

For in-domain data, the **Span** features alone resulted in a performance gain slightly larger than **TWNC**. However, they seemed too domain-specific as they did not work well for out-of-domain data. By contrast, the **Web span** features brought consistent gains to both in- and out-of-domain data. Adding only the **Paraphrase** features to the **Base** model had a negative impact for in-domain data and out-of-domain 1.

The results indicate the complementary nature of the edge, span and paraphrase features. The combination of these features generally boosted performance. This was especially true for out-of-domain data. 0.63% and 0.70% gains were obtained when the **Paraphrase** features were added to the **Base** + **Span** + **Web span** model even though **Paraphrase** alone did not work well. **TWNC** alone worked well for out-of-domain data, but further gain was obtained when the **Span**, **Web span** and **Paraphrase** features were added. The highest scores were achieved by the **Base** + **Span** + **Web span** + **TWNC** + **Paraphrase** model (hereafter, the full model) for all datasets.

4.5 Discussion

We found that web span features were useful for complementing weak edges. In "[*fonon* [*jiyuu enerugī*]]" ([phonon [free energy]]), for example, the well-known term "*jiyuu enerugī*" (free energy) is modified ad hoc by "*fonon*" (phonon). The edge between "*fonon*" (phonon) and "*enerugī*" was so weak that in the **Base** model it was unable to override the strong preference for short-distance dependency. On the other hand, the web span feature strongly supported "*jiyuu enerugī*" (free energy). A powerful span means that non-head words within the span must not be modified from outside the span. For this reason, the edge between "*fonon*" (phonon) and "*jiyuu*" (free) was ruled out.

However, the span features sometimes had an adverse impact on parsing. Consider the following example.

[[*shinka*	*gēmu*]	*riron*]
evolution	game	theory
evolutionary game theory.		

The full model wrongly output "[*shinka* [*gēmu riron*]]" ([evolutionary [game theory]]). This noun compound can be interpreted as a fusion of "*shinka gēmu*" (evolutionary game) and more prominent "*gēmu riron*" (game theory). In other words, the non-head word (game) of the latter is modified by the grafting of the former. However, our span features do not allow such an operation.

This inherent weakness of span might also explain the poor performance of paraphrase features. We investigated the weight vector of the full model. Somewhat surprisingly, we found that among four paraphrase features, only $c_{\text{PARA}}(n_j, {}_{j+1}S_k)$ (the argument's head word and the predicate's span) had a positive weight. In other words, the argument span ${}_iS_j$ was considered useless or even harmful by the model. One possible reason is the productivity of the argument. ${}_iS_j$ in paraphrase features imposes the condition that n_j must not be modified from outside the span. However, such a modification occurs very often. Even if the correct pair ${}_iS_j, {}_{j+1}S_k$ is covered by the web corpus, it is often blocked by another pair ${}_{i'}S_j, {}_{j+1}S_k$ $(i < i')$, which usually has larger counts.

While we use flat spans, the same is true of genuine tree models. Our span features show some similarity to adaptor grammars (Johnson et al., 2007), a generalization of probabilistic context-free grammars.[8] An adaptor grammar directly considers a distribution of subtrees rooted by a common non-terminal instead of decomposing them into derivation rules. A subtree is completely expanded into terminals. If subtrees are collapsed into terminal sequences, they become spans.

The adaptor grammar does not allow subtrees to be modified partially. This is unfavorable in general because "*parallel processing*" is productively modified by an adverb and transformed into "*massively parallel processing*" for example. To address this problem, we need to handle incompletely expanded trees that are to be completed by a *substitution* operator and/or we need to introduce an *insertion* operator (Shindo et al., 2011). For the semi-supervised setting of noun compound analysis, we may need collapsed versions of these operations.

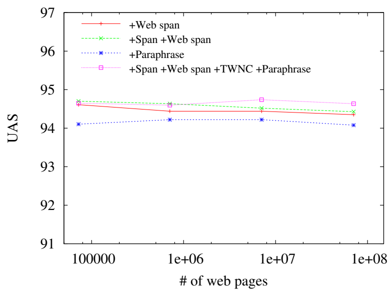
4.6 Effect of corpus size

Finally, we investigated the effect of the size of the corpus from which we calculated web statistics. We reduced the number of web pages to 1/10, 1/100 and 1/1000. The models were trained and tested as before.

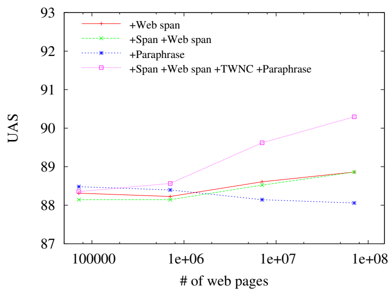
Figures 5(a)-(c) show UASs in relation to corpus size. In-domain data did not receive benefit from the increase of unannotated text. It seems that the Base and Span features, which were learned directly from annotated data, were too informative for the other features to work with.

For out-of-domain data, accuracy largely consistently improved with the corpus size except for the **Base** + **Paraphrase** model. The graphs indicate that further gain can be achieved by

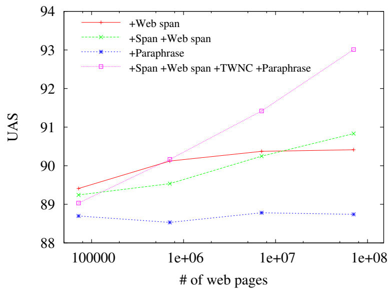
[8]Adaptor grammars are not irrelevant to dependency parsing as dependency grammars can be transformed into context-free grammars (Johnson, 2007). The original adaptor grammars do not allow self-recursion, which is integral to dependency-derived CFGs, but this restriction was overcome by a variational inference scheme (Cohen et al., 2010).



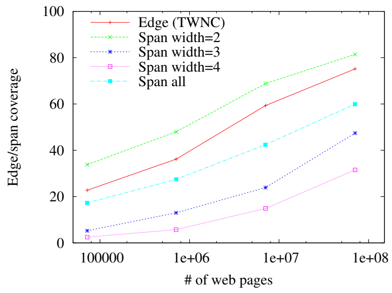
(a) In-domain.



(b) Out-of-domain 1.



(c) Out-of-domain 2.



(d) Out-of-domain 1.

Figure 5: Effect of corpus size. (a)-(c) Unlabeled attachment scores in relation to corpus size. (d) Edge/span coverage in relation to corpus size.

simply enlarging the corpus. This is supported by Figure 5d, which depicts how many edges and spans in test data appear at least once in the web statistics. There is much room for improving coverage.

Conclusion

In this paper, we proposed a semi-supervised method for noun compound analysis that combined span features with edge features. Experiments show that span features improve accuracy and that further gain is obtained when they are combined with edge features.

Words within noun compounds are arranged in a rather fixed order, and adding a word in between appears to impair semantic coherence. The very fact that people often choose bracketing to denote the internal structure of a noun compound may be an intuitive justification of the use of spans since the bracket structure obscures dependency but shows spans more clearly.

Acknowledgment

This work was partly supported by JST CREST.

References

Banko, M. and Brill, E. (2001). Mitigating the paucity-of-data problem: exploring the effect of training corpus size on classifier performance for natural language processing. In *Proceedings of the First International Conference on Human Language Technology Research (HLT 2001)*, pages 1–5.

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

Barker, K. (1998). A trainable bracketer for noun modifiers. In *Proceedings of the 12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, pages 196–210.

Bergsma, S., Pitler, E., and Lin, D. (2010). Creating robust supervised classifiers via web-scale N-gram data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 865–874.

Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J. (2007). Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, pages 858–867.

Cohen, S. B., Blei, D. M., and Smith, N. A. (2010). Variational inference for adaptor grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 564–572.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Eisner, J. (1996). The new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, pages 340–345.

Eisner, J. and Satta, G. (1999). Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 457–464.

Flannery, D., Miyao, Y., Neubig, G., and Mori, S. (2011). Training dependency parsers from partially annotated corpora. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 776–784.

Johnson, M. (2007). Transforming projective bilexical dependency grammars into efficiently-parsable CFGs with unfold-fold. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 168–175.

Johnson, M., Griffiths, T. L., and Goldwater, S. (2007). Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models. In *Advances in Neural Information Processing Systems*, volume 19, pages 641–648.

Kageura, K., Yoshioka, M., Takeuchi, K., Koyama, T., Tsuji, K., Yoshikane, F., and Okada, M. (1999). Overview of TMREC tasks. In *Proceedings of the First NTCIR Workshop on Research in Japanese Text Retrieval and Term Recognition*, page 415.

Kaji, N. and Kitsuregawa, M. (2011). Splitting noun compounds via monolingual and bilingual paraphrasing: A study on Japanese Katakana words. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 959–969.

Kawahara, D. and Kurohashi, S. (2001). Japanese case frame construction by coupling the verb and its closest case component. In *Proceedings of the First International Conference on Human Language Technology Research (HLT 2001)*, pages 204–210.

Kawahara, D. and Kurohashi, S. (2006). Case frame compilation from the web using high-performance computing. In *Proceedings of The 5th International Conference on Language Resources and Evaluation (LREC-06)*, pages 1344–1347.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 48–54.

Koo, T. and Collins, M. (2010). Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11.

Kudo, T. and Matsumoto, Y. (2002). Japanese dependency analysis using cascaded chunking. In *Proceedings of the 6th Conference on Natural Language Learning*, pages 1–7.

Lapata, M. and Keller, F. (2004). The web as a baseline: Evaluating the performance of unsupervised web-based models for a range of NLP tasks. In *HLT-NAACL 2004: Main Proceedings*, pages 121–128.

Lauer, M. (1995). Corpus statistics meet the noun compound: Some empirical results. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 47–54.

Liberman, M. and Sproat, R. (1992). The stress and structure of modified noun phrases in English. In Sag, I. A. and Szabolcsi, A., editors, *Lexical Matters*, pages 131–181. Center for the Study of Language.

Marcus, M. P. (1980). *Theory of Syntactic Recognition for Natural Languages*. MIT Press.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

McDonald, R., Crammer, K., and Pereira, F. (2005). Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98.

McDonald, R. and Pereira, F. (2006). Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88.

Nakagawa, H. and Mori, T. (2002). A simple but powerful automatic term extraction method. In *COLING-02 on COMPUTERM 2002: Second International Workshop on Computational Terminology - Volume 14*, pages 29–35.

Nakov, P. and Hearst, M. (2005a). Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 17–24.

Nakov, P. and Hearst, M. (2005b). Using the web as an implicit training set: Application to structural ambiguity resolution. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 835–842.

Pitler, E., Bergsma, S., Lin, D., and Church, K. (2010). Using web-scale N-grams to improve base NP parsing performance. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 886–894.

Post, M. and Gildea, D. (2009). Bayesian learning of a tree substitution grammar. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 45–48.

Pustejovsky, J., Bergler, S., and Anick, P. (1993). Lexical semantic techniques for corpus analysis. *Computational Linguistics*, 19(2):331–358.

Resnik, P. S. (1993). *Selection and Information: A Class-Based Approach to Lexical Relationships*. PhD thesis, University of Pennsylvania.

Sasano, R., Kawahara, D., and Kurohashi, S. (2009). The effect of corpus size on case frame acquisition for discourse analysis. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 521–529.

Shindo, H., Fujino, A., and Nagata, M. (2011). Insertion operator for bayesian tree substitution grammars. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 206–211. Association for Computational Linguistics.

Uchimoto, K., Sekine, S., and Isahara, H. (1999). Japanese dependency structure analysis based on maximum entropy models. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 196–203.

Vadas, D. and Curran, J. (2007a). Adding noun phrase structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 240–247.

Vadas, D. and Curran, J. (2007b). Large-scale supervised models for noun phrase bracketing. In *Proceedings of the Conference of the Pacific Association for Computational Linguistics (PACLING)*, pages 104–112.

Wood, F., Gasthaus, J., Archambeau, C., James, L., and Teh, Y. W. (2011). The sequence memoizer. *Communications of the Association for Computing Machines*, 54(2):91–98.