A Generative Dependency N-gram Language Model: Unsupervised Parameter Estimation and Application

Chenchen Ding[†] and Mikio Yamamoto[†]

We design a language model based on a generative dependency structure for sentences. The parameter of the model is the probability of a *dependency N-gram*, which is composed of lexical words with four types of extra tag used to model the dependency relation and valence. We further propose an unsupervised expectation-maximization algorithm for parameter estimation, in which all possible dependency structures of a sentence are considered. As the algorithm is language-independent, it can be used on a raw corpus from any language, without any part-of-speech annotation, tree-bank or trained parser. We conducted experiments using four languages, i.e., English, German, Spanish and Japanese, to illustrate the applicability and the properties of the proposed approach. We further apply the proposed approach to a Chinese microblog data set to extract and investigate Internet-based, non-standard lexical dependency features of user-generated content.

Key Words: *N-gram language model, Generative dependency structure, Unsupervised algorithm, Microblog*

1 Introduction

Statistical language models are a fundamental component of speech recognition systems, machine translation systems and so forth. At present, the N-gram language model is the most widely used approach. This model focuses on sequences of neighboring lexical words (Figure 1) and uses the probabilities of these sequences as model parameters. Due to the complete lexicalization of the N-gram language model, local features of word sequences can be well modeled. However, an N-gram language model cannot capture relatively long-range features, because it considers a sentence as a flat string and ignores its structure.

Revealing a sentence structure is the task of parsing, which is based on linguistically oriented formulations, and it focuses on generating the likeliest structure for a given sentence, using constituency- or dependency-based formulations. The former organizes continuous word sequences in a hierarchy of small to large range groups with linguistically oriented labels, while the latter directly links words with dependency relations[1] (Figure 2).

[†] Department of Computer Science, University of Tsukuba

[1] In general, the dependency relations can be further classified using linguistically oriented labels. However, they are not indispensable, and we do not use them in our approach. For the figures in this study, we use the following representation to show the dependency structure. If two aligned words are on different levels, the upper one is the head of the lower one; if they are on the same level, they are siblings.

In this study, we focus on introducing sentence structure into language modeling. We propose a generative dependency N-gram language model that integrates the generative dependency structure of a sentence into the original N-gram language model. We prefer the dependency-based formulation because it can directly model the relations between words. In the proposed model, the parameter is the probability of the dependency N-gram, which is a sequence of words along the dependency structure rather than along a flat left-to-right string. The proposed model is thus as completely lexical as the original N-gram language model. We further propose an expectation-maximization (EM) algorithm for estimating the probability of arbitrary order[2] dependency N-grams, by considering all possible dependency structures[3] of a sentence (Figure 3).

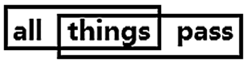

Fig. 1 N-gram language model. For $N = 2$, the model treats the English sentence "*all things pass*" as being constructed from the bi-grams *(all, things)* and *(things, pass)*.

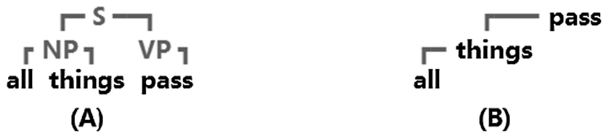

Fig. 2 Constituency-based parsing **(A)** and dependency-based parsing **(B)** of the English sentence "*all things pass*".

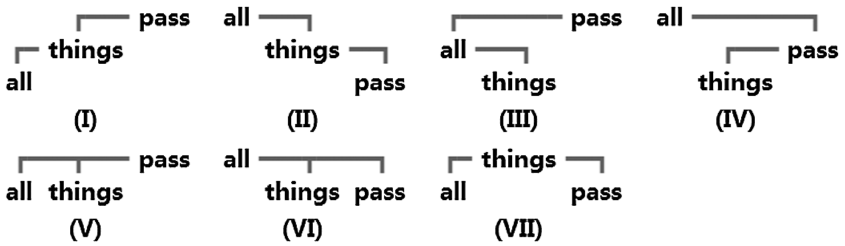

Fig. 3 All possible dependency structures for the English sentence "*all things pass*". **(I)** is the linguistically correct structure, while the original N-gram language model handles the sentence as if it has the structure labeled **(II)**. We consider all these structures in our unsupervised estimation algorithm.

[2] In this study, the term "order" of a dependency N-gram means the number of lexical words (N) in a head-modifier chain, which is used as an extension of the original N-gram. In the context of dependency parsing, "order" generally means the number of words in a treelet, which can contain relations such as siblings, ancestors and descendants. That is, the "order" in this study is restricted to include only ancestors.

[3] Only projective dependency structures are considered.

The proposed algorithm is unsupervised, language-independent and needs no linguistic information.

2 Related Work

The technical report by (Chen and Goodman 1998) has compared various approaches to the N-gram language model and the modified Kneser-Ney (KN) discounting proposed in it is still the state-of-the-art. Since the N-gram language model only captures local lexical features, there have been proposals to generalize the lexical N-gram by word class (Brown, deSouza, Mercer, Pietra, and Lai 1992) or to model long-range word co-occurrences by word triggers (Tillmann and Ney 1997). However, these models are unaware of the sentence structure, and they basically take a sentence as a flat string.

Many approaches have been proposed for constituency-based parsing (Collins 1998; Klein and Manning 2003, 2004) and for dependency-based parsing (Eisner 1996; Lee and Choi 1997; Kudo and Matsumoto 2002; Klein and Manning 2004; Nivre 2008; Koo and Collins 2010; Zhang and McDonald 2012). Discriminative approaches (Kudo and Matsumoto 2002; Nivre 2008) are used more than generative ones for dependency-based parsing, because a generative model is usually restricted to being bi-lexical (i.e., the components are bi-grams of head-modifier pairs). Specific algorithms have been designed to handle more complex dependency relations (Koo and Collins 2010; Zhang and McDonald 2012), and these allow the consideration of more lexical information in a generative model.

There have been attempts to integrate sentence structure into language modeling. (Chelba and Jelinek 2000) have proposed a constituency-based approach, but the use of language-dependent non-terminals cannot be avoided. There are also dependency-based approaches. One straightforward method is to construct a language model based on the decisively best structure produced by a parser (Stolcke, Chelba, Engle, Jimenez, Mangu, Printz, Ristad, Rosenfeld, Wu, Jelinek, and Khudanpur 1997; Gao and Suzuki 2003; Graham and van Genabith 2010). These approaches can be considered to convert the left-to-right string in the original N-gram model to a completely syntax-driven tree structure. A more reasonable method is to consider all dependency structures of a sentence. One such attempt is the bi-gram head-modifier model of (Lee and Choi 1998), which is based on the parsing approach of (Lee and Choi 1997).

In our approach, we consider all dependency structures of a sentence and try to include more lexical information. We extend the approach of (Lee and Choi 1997) to head-modifier chains of arbitrary words, rather than head-modifier pairs. We also use extra tags of the type typically found in parsing models. These tags are treated as general lexical words and are used to model the valence of a head word. The parsing approaches of (Koo and Collins 2010) also handle more lexical information in a dependency structure, including complex relations, such as the sibling relation. However, the use of high-order dependency patterns in the approach is limited. As described in (Zhang and McDonald 2012), arbitrary orders of lexical information with arbitrary dependency relations can be handled only if the proper algorithms are designed, and designing these becomes more complex with the increasing number of lexical words and dependency relations. Our approach concentrates only on the head-modifier chain, that is, a sequence of parent-child relations. Therefore, our approach is a direct extension of the original N-gram model, which models a lexical word sequence, without branching. We will also show that a head-modifier chain of arbitrary order can be modeled in a uniform algorithm, which will not increase in formulation complexity when the order increases.

3 Generative Dependency Model

We model the marginal probability of a sentence S over set D of all possible dependency structures of S: $P(S) = \sum_{d \in D} P(S, d)$. As described in (Klein and Manning 2004), if we separate the dependency structure and lexicalization, then $\sum_{d \in D} P(S, d) = \sum_{d \in D} P(d)P(S|d)$. The term $P(S|d)$ is given by a model of completely lexical word sequences with dependency relations. However, the term $P(d)$, which is the probability of a dependency graph without lexical words, is difficult to model. In earlier studies , the $P(d)$ term is taken as a constant or omitted (i.e., taken as 1) for simplicity, as in (Paskin 2002; Lee and Choi 1998). For example, in (Lee and Choi 1998), the probability of a sentence S is calculated as $P(S) = \sum_{d \in D} \prod_{(x \to y) \in d} P(x \to y)$, where the element $(x \to y)$ is a lexical head-modifier pair in a given dependency structure d. Thus, the term $\prod_{x \to y \in d} P(x \to y)$ is equivalent to $P(S|d)$.

To combine the dependency structure and lexicalization, the *valence*, which represents the modifier numbers of a head word, should be modeled. A *dependency model with valence* (DMV) is proposed by (Klein and Manning 2004). DMV is a generative model that includes a special mark, *STOP*, to terminate the modifier sequence of a head word. With the help of the *STOP* mark, the number of modifiers can be controlled. It is necessary to distinguish the two types of parameters, i.e., P_{STOP} and P_{CHOOSE} in the bi-gram estimation, which makes it difficult to extend the approach to higher orders.

In a similar approach to that used in DMV, we introduce four types of tag to normalize the distribution of modifier numbers (the valence) of a head word. In this study, we use $\langle L \rangle$

(resp. $\langle /L \rangle$) and $\langle R \rangle$ (resp. $\langle /R \rangle$) to show the start (resp. end) of the left and right modifier word sequences of a head word. The dependency structure can thus be organized as nested word sequences. Specifically, modifier word sequences of a head word are of the form $M = m_0^{\phi+1} \equiv m_0, m_1, \cdots, m_{\phi+1}$, where $m_0 \equiv \langle O \rangle$, $m_{\phi+1} \equiv \langle /O \rangle$ ($O \in \{L, R\}$), and m_1^{ϕ} is a lexical ϕ-word sequence. We show an example of the dependency structure in Figure 4. For example, in Figure 4, the word *get* has a left modifier word sequence "$\langle L \rangle$ *i* $\langle /L \rangle$" and a right modifier word sequence "$\langle R \rangle$ *book from* $\langle /R \rangle$". In contrast to the treatment in DMV, we treat tags as ordinary words in the parameter estimation. This means parameters of our model have a uniform representation, by which our approach can be easily extended to arbitrary high orders.

Our model is essentially equivalent to the generative *Model C* in (Eisner 1996). In other words, the sequence $\langle O \rangle m_1^{\phi} \langle /O \rangle$ ($O \in \{L, R\}$) is generated as a Markov sequence to serve as the modifier word sequences (left/right separately) of the head word. The "start tag" $\langle O \rangle$ always satisfies $P(m_0 = \langle O \rangle) \equiv 1$ to represent the nested structure. The "end tag" $\langle /O \rangle$ terminates the generation process: the larger $P(m_{\phi+1} = \langle /O \rangle)$ is, the smaller ϕ, which is the number of generated words, becomes and vice versa.[4]

Without loss of generality, the probability of $m_{\kappa+1}$ ($0 < \kappa \leq \phi$) in $M = m_0^{\phi+1}$ can be represented by $P(m_{\kappa+1} | m_0^{\kappa}, H)$, where H is the history of M along the generated path[5]. We use

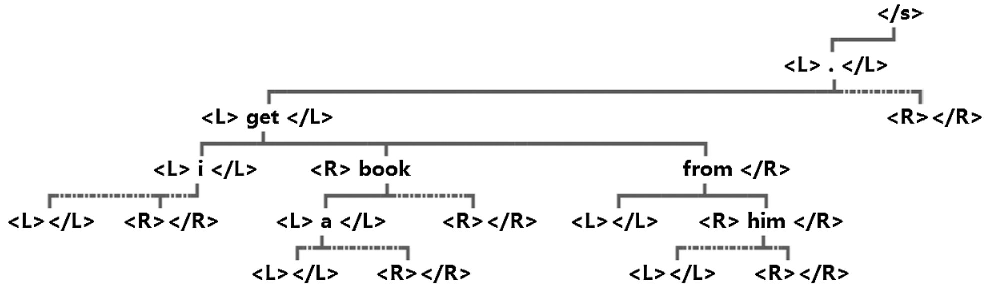

Fig. 4 A dependency structure for the English sentence "*i get a book from him .*", with $\langle L \rangle$, $\langle R \rangle$, $\langle /L \rangle$, $\langle /R \rangle$ tags. The root of the sentence is marked as $\langle /s \rangle$, and for a word without modifiers, its modifier word sequences are $\langle O \rangle \langle /O \rangle$, where $O \in \{L, R\}$ (marked by dashed lines).

[4] As to the consistency of our language model, that is, whether $\sum_{S \in L} P(S) = 1$ for every possible sentence S in a language L, we note that it **cannot** be guaranteed by the generative structure alone. As discussed in (Wetherell 1980), a language generated by a probabilistic context-free grammar cannot be guaranteed to be consistent, because the generation process cannot be guaranteed to finish, even when the probabilities are normalized. However, in this situation, the probabilities of terminal sequences (i.e. sentences) will be very low, which will lead to a poor performance of the language model. Thus, the results for the proposed approach reported in this study may underestimate probabilities but will not overestimate them.

[5] The generation process can be realized in a depth-first or a breadth-first way, but distinction is not essential.

the independent assumption that the probability of a word in the generation process only depends on its direct ancestors and the orientation between them. The probability can be simplified to:

$$P(h^0|o^1, h^1, \ldots, o^{n-1}, h^{n-1}), \tag{1}$$

where h^k ($k \in [1, n-1]$) is the head word of h^{k-1} and o^k ($k \in [1, n-1]$) shows the orientation between them. Specifically, h^k ($k \in [0, n-1]$) can be any of the following:

- a lexical word in a given sentence,
- a $\langle/O\rangle$ ($O \in \{L, R\}$) tag,
- the sentence-ending tag $\langle/s\rangle$, which is taken as the root of a sentence, or
- the sentence-beginning tag $\langle s\rangle$, which is taken as a trivial placeholder,

and o^k ($k \in [1, n-1]$) is a $\langle O\rangle$ ($O \in \{L, R\}$) tag.

The "slash" tags, $\langle/s\rangle$, $\langle/L\rangle$ and $\langle/R\rangle$, are taken as lexical words, which are represented by h^k. The "no-slash" tags, $\langle L\rangle$ and $\langle R\rangle$, show the direction of a modifier word against its head word, which is represented by o^k. Specifically, for $k \in [1, n-1]$, we have $o^k = \langle L\rangle$ when h^{k-1} is on the left side of h^k, and $o^k = \langle R\rangle$ when h^{k-1} is on the right side of h^k.[6] Further, the sentence-beginning tag $\langle s\rangle$ is used as a trivial placeholder to increase the order of Exp. (1) to n. It is used only when $h^k = \langle/s\rangle$ for some $k < n-1$.[7] For the tags, there are some noticeable properties, such as the following:

- if $h^k = \langle/O\rangle$ ($O \in \{L, R\}$), then $k \equiv 0$; because $\langle/O\rangle$ ($O \in \{L, R\}$) cannot have modifiers,
- if $h^0 = \langle/L\rangle$, then $o^1 \equiv \langle L\rangle$, and if $h^0 = \langle/R\rangle$, then $o^1 \equiv \langle R\rangle$,
- if $h^k = \langle/s\rangle$ ($k \in [1, n-1]$), then $o^k \equiv \langle L\rangle$,
- if $h^k = \langle/s\rangle$ ($k \in [1, n-2]$), then both $o^{k+1} \equiv \langle R\rangle$ and $h^{k+1} = \langle s\rangle$,
- if $h^k = \langle s\rangle$ ($k \in [1, n-2]$), then both $o^{k+1} \equiv \langle R\rangle$ and $h^{k+1} = \langle s\rangle$.

For example, a dependency N-gram is ($\langle/L\rangle$, $\langle L\rangle$, *him*, $\langle R\rangle$, *from*, $\langle R\rangle$, *get*, $\langle L\rangle$, ., $\langle L\rangle$, $\langle/s\rangle$) in the dependency structure illustrated in Figure 4. We can see $\langle O\rangle$ ($O \in \{L, R\}$) tags between words show the relevant position between head and modifier words. In fact, all words in a modifier sequence share the same $\langle O\rangle$ ($O \in \{L, R\}$). For example, in Figure 4, the "*book*" and "*from*" share the same $\langle R\rangle$ tag as they are both in the right modifier word sequence of the head word "*get*".

The sequence $(h^0, o^1, h^1, \ldots, o^{n-1}, h^{n-1})$ in Exp. (1) is referred as a dependency N-gram.

[6] The use of an index in Exp. (1) can be interpreted as h^0 is on the o^1 side of its head word h^1, which is on the o^2 side of its head word h^2, and continue this pattern to h^{n-2}, which is on the o^{n-1} side of its head word h^{n-1}.

[7] Hence, we omit the $\langle s\rangle$ tag in Figure 4 as it adds nothing to the structure.

Exp. (1) is the probability of the dependency N-gram and thus the parameter of our model, where the dependency relation and valence are modeled uniformly for arbitrary orders.

From the probability of the dependency N-gram of Exp. (1), the probability of a given dependency structure d of a sentence S can be calculated as $\prod_{h^0 \in d} p(h^0|o^1, h^1, \ldots, o^{n-1}, h^{n-1})$. Because of how we use $\langle /O \rangle$ ($O \in \{L, R\}$) tags, the $\prod_{h^0 \in d} p(h^0|o^1, h^1, \ldots, o^{n-1}, h^{n-1})$ is equivalent to $P(S, d)$ rather than to $P(S|d)$, as in (Lee and Choi 1998). We show an example of the $P(S, d)$ from Exp. (2) to Exp. (6) according to the structure in Figure 4, proceeding layer by layer[8]. We can see there are many terms of the type $h^0 = \langle /O \rangle$ ($O \in \{L, R\}$) in the calculation. This can be considered a "discount" for the product of lexical terms to represent the "structure probability" $p(d)$, although $p(d)$ is never separated as an individual term because we merge the lexicalization and dependency structure in our calculations.

$$
\begin{aligned}
&P(S,d) = \\
&\quad P(.|\langle L\rangle\ \langle /s\rangle) && (2)\\
&\quad \cdot P(get|\langle L\rangle\ .\ \langle L\rangle\ \langle /s\rangle)P(\langle /L\rangle|\langle L\rangle\ .\ \langle L\rangle\ \langle /s\rangle)P(\langle /R\rangle|\langle R\rangle\ .\ \langle L\rangle\ \langle /s\rangle) && (3)\\
&\quad \cdot P(i|\langle L\rangle\ get\ \langle L\rangle\ .\cdots)P(\langle /L\rangle|\langle L\rangle\ get\ \langle L\rangle\ .\cdots)\\
&\qquad \cdot P(book|\langle R\rangle\ get\ \langle L\rangle\ .\cdots)P(from|\langle R\rangle\ get\ \langle L\rangle\ .\cdots)P(\langle /R\rangle|\langle R\rangle\ get\ \langle L\rangle\ .\cdots) && (4)\\
&\quad \cdot P(\langle /L\rangle|\langle L\rangle\ i\ \langle L\rangle\ get\cdots)P(\langle /R\rangle|\langle R\rangle\ i\ \langle L\rangle\ get\cdots)\\
&\qquad \cdot P(a|\langle L\rangle\ book\ \langle R\rangle\ get\cdots)P(\langle /L\rangle|\langle L\rangle\ book\ \langle R\rangle\ get\cdots)P(\langle /R\rangle|\langle R\rangle\ book\ \langle R\rangle\ get\cdots)\\
&\qquad \cdot P(\langle /L\rangle|\langle L\rangle\ from\ \langle R\rangle\ get\cdots)P(him|\langle R\rangle\ from\ \langle R\rangle\ get\cdots)P(\langle /R\rangle|\langle R\rangle\ from\ \langle R\rangle\ get\cdots) && (5)\\
&\quad \cdot P(\langle /L\rangle|\langle L\rangle\ a\ \langle L\rangle\ book\cdots)P(\langle /R\rangle|\langle R\rangle\ a\ \langle L\rangle\ book\cdots)\\
&\qquad \cdot P(\langle /L\rangle|\langle L\rangle\ him\ \langle R\rangle\ from\cdots)P(\langle /R\rangle|\langle R\rangle\ him\ \langle R\rangle\ from\cdots) && (6)
\end{aligned}
$$

The probability of a sentence S can then be calculated by $P(S) = \sum_{d \in D} P(S, d)$, where the left-to-right generation of the original N-gram model is naturally included, and the probability of it will be discounted by the terms of the form $h^0 = \langle /O \rangle$ ($O \in \{L, R\}$).

4 Parameter Estimation

4.1 Notation

We denote a sentence with l words as $S = w_0^{l+1} \equiv w_0, w_1, \cdots, w_{l+1}$, where $w_0 \equiv \langle s \rangle$ and $w_{l+1} \equiv \langle /s \rangle$; each w_k ($k \in [1, l]$) is an ordinary lexical word. In a sentence $S = w_0^{l+1}$, we denote a dependency N-gram $(h^0, o^1, h^1, \ldots, o^{n-1}, h^{n-1})$ by an N-tuple $\mathbf{d} = (d_0, d_1, \ldots, d_{n-1})$ according to the following definition.

$$
d_k = \begin{cases} h^k, \text{ if } k = 0 \text{ and } h^k \textit{ is a } \langle /O \rangle\ (O \in \{L, R\}) \\ i \text{ such that } h^k = w_i, \text{ otherwise} \end{cases} \tag{7}
$$

[8] From Exp. (4), we omit part of the histories for brevity.

The definition of d_k in Exp. (7) thus shows the relation between h^k and d_k. Because the $\langle s \rangle$, $\langle /s \rangle$, $\langle /L \rangle$ and $\langle /R \rangle$ tags are taken as lexical words in a dependency N-gram, they can appear in a $\mathbf{d}$. In our notation, $\langle s \rangle$ and $\langle /s \rangle$ are assigned absolute positions of 0 and $l+1$, respectively, in an l-word sentence, so they can be trivially integrated in a $\mathbf{d}$. Conversely, as $\langle /L \rangle$ and $\langle /R \rangle$ tags are attached to every word in a sentence, we cannot assign absolute positions to them, so, they remain in a $\mathbf{d}$ with no transformation to absolute position. Consequently, d_k in a $\mathbf{d}$ can be an integer in $[0, l+1]$ or a $\langle /O \rangle$ ($O \in \{L, R\}$) tag. In fact, because $\langle /L \rangle$ and $\langle /R \rangle$ tags can appear only as h^0 in a dependency N-gram, they only appear as d_0 in a $\mathbf{d}$. The N-tuple $\mathbf{d}$ with a $d_0 = \langle /O \rangle$ ($O \in \{L, R\}$) will play a special role in the recursive definition in Section 4.2.

Because the magnitudes of d_k and d_{k+1} ($k \in [0, n-2]$) show the orientation, o^{k+1} can be omitted. In addition, o^{k+1} can be unambiguously omitted for the $\langle /L \rangle$ and $\langle /R \rangle$ tags because of the properties we mentioned in the previous section. Consequently, the $\langle L \rangle$ and $\langle R \rangle$ tags never need to appear in a $\mathbf{d}$. As an example, the dependency N-gram ($\langle /L \rangle$, $\langle L \rangle$, *him*, $\langle R \rangle$, *from*, $\langle R \rangle$, *get*, $\langle L \rangle$, *.*, $\langle L \rangle$, $\langle /s \rangle$) in the dependency structure illustrated in Figure 4 can be denoted by a $\mathbf{d} = (\langle /L \rangle, 6, 5, 2, 7, 8)$ given the sentence "*i (1) get (2) a (3) book (4) from (5) him (6) . (7)*".

(Lee and Choi 1997) propose the *complete-link set* and *complete-sequence set* for head-modifier pairs (i.e., a dependency bi-gram in our model) to handle all possible projective dependency structures of a sentence in a recursive manner. We follow the terms they use and extend their definitions to adapt them to our dependency N-gram model. We use $Link(\mathbf{d})$ to denote the complete-link set of an N-tuple $\mathbf{d}$ and $Seq(\mathbf{d})$ for the complete-sequence set.

In (Lee and Choi 1997), the complete-link set of a span $[i, j]$ in a sentence is composed of all possible dependency structures within the span, with the directional dependency link of the two words w_i and w_j. The complete-sequence set of a span $[i, j]$ is defined as the set of all possible sequences with any number (including zero) of adjacent complete-link sets having the same direction within the span. By our notation, the word at d_1 is the direct head of the word at d_0 for $Link(d_0, d_1)$, but the word at d_1 is an ancestor (not only a direct head) of the word at d_0 for $Seq(d_0, d_1)$. The two types of set can be defined recursively, and the set of all possible dependency structures of a sentence $S = w_0^{l+1}$ is the complete-sequence set over the span $[1, l+1]$ or is the complete-link set over the span $[0, l+1]$[9]. We illustrate these recursive relations in Figure 5 and 6.[10]

[9] Because $p(\langle /s \rangle | \langle R \rangle\ \langle s \rangle) \equiv 1$, which is one of the properties we have described, the two complete sets have the same probability. This is also mentioned in (Lee and Choi 1997).

[10] By our notation, there is no necessity to show the head-modifier relation by arrows in Figure 5 to 8. These figures show the relation between N-tuples, rather than directional pairs.

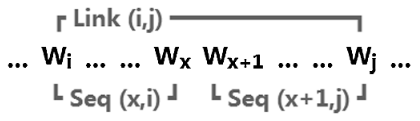

Fig. 5 *Link*$(\mathbf{d} = (i, j))$. In (Lee and Choi 1997), for a span $[i, j]$, *Link*(i, j) is composed of the dependency link of w_i and w_j, and all possible pairs of complete-sequence sets *Seq*(x, i) and *Seq*$(x + 1, j)$.

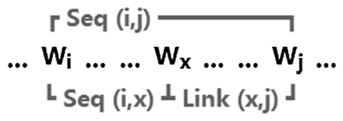

Fig. 6 *Seq*$(\mathbf{d} = (i, j))$. In (Lee and Choi 1997), for a span $[i, j]$, *Seq*(i, j) is composed of all possible pairs of complete-sequence set *Seq*(i, x) and complete-link set *Link*(x, j).

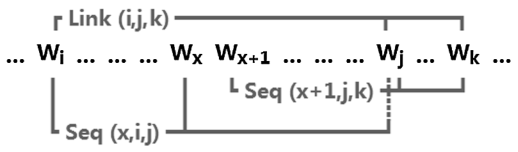

Fig. 7 *Link*$(\mathbf{d} = (i, j, k))$. In our model, an extended high-order (three-order is shown here) complete-link set *Link*(i, j, k) is composed of the N-tuple $\mathbf{d}$, and all possible pairs of complete-sequence sets *Seq*(x, i, j) and *Seq*$(x + 1, j, k)$.

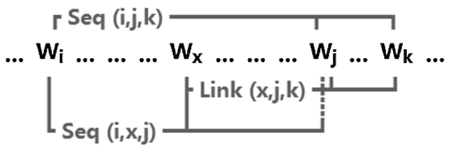

Fig. 8 *Seq*$(\mathbf{d} = (i, j, k))$. In our model, an extended high-order (three-order is shown here) complete-sequence set *Seq*(i, j, k) is composed of all possible pairs of complete-sequence set *Seq*(i, x, j) and complete-link set *Link*(x, j, k).

Because more than two words are involved in the proposed dependency N-gram, we generalize the two types of set for the N-tuples $\mathbf{d}$, rather than just spans. The generalization still retains the properties of d_0 and d_1 in *Link*$(\mathbf{d})$ and *Seq*$(\mathbf{d})$, as well as the recursive properties of the two types of set. We show the examples of a dependency tri-gram in Figure 7 and 8.

4.2 Recursive Definition

Here, we provide the formulation of the recursive definition of the complete-link set and complete-sequence set for an arbitrary order dependency N-gram.

In the description of the calculation example shown from Exp. (2) to Exp. (6) in Section 3, we mentioned that those terms $h^0 = \langle /O \rangle$ $(O \in \{L, R\})$ can be considered as a "discount" of the product of lexical terms. By the definition in Exp. (7) in Section 4.1, we can further see that $h^0 = \langle /O \rangle$ $(O \in \{L, R\})$ terms are represented by the N-tuple $\mathbf{d}$ with $d_0 = \langle /O \rangle$ $(O \in \{L, R\})$ and the other lexical terms are represented by the N-tuple $\mathbf{d}$ in which all the d_k $(k \in [0, n-1])$ are integers. For clarity, in this section, we will first describe the recursion definition of lexical terms without $d_0 = \langle /O \rangle$ $(O \in \{L, R\})$ involved up to Exp. (17). Next, we turn to the "discount" terms, that is, the case of $d_0 = \langle /O \rangle$ $(O \in \{L, R\})$, from Exp. (18) to Exp. (21).

First, due to the properties of the projective dependency structure, any d_k ($k \in [1, n-1]$) in the N-tuple $\mathbf{d} = (d_0, d_1, \ldots, d_{n-1})$ needs to satisfy the following constraint of Exp. (8) to guarantee that a head word is outside the range covered by a chain of its descendants.

$$d_k > \max(d_0, \cdots, d_{k-1}), \text{ or } d_k < \min(d_0, \cdots, d_{k-1}) \tag{8}$$

The max(·) and min(·) operations are used to get the maximum and minimum from a tuple composed of integers.

Trivially, we take $\langle /s \rangle$ as the *root* mark of a sentence $S = w_0^{l+1}$, and the $\langle s \rangle$ as the head of itself or as the head of the $\langle /s \rangle$. So, we have the following constraints:

$$\text{if } d_{k-1} = l+1 \text{ or } d_{k-1} = 0, \text{then } d_k = 0, \text{for } k \in [1, n-1] \tag{9}$$

To reveal the relations between N-tuples, we introduce three basic operations, *Push*, *Cover* and *Insert*, over an index x (absolute word position) and an N-tuple $\mathbf{d} = (d_0, d_1, \ldots, d_{n-1})$:

$$\mathit{Push}(x, \mathbf{d}) = (x, d_0, d_1, \ldots, d_{n-2}) \tag{10}$$

$$\mathit{Cover}(x, \mathbf{d}) = (x, d_1, d_2, \ldots, d_{n-1}) \tag{11}$$

$$\mathit{Insert}(x, \mathbf{d}) = (d_0, x, d_1, \ldots, d_{n-2}) \tag{12}$$

With the three operations, we can express the relation shown in Figure 7 as follows:

$$\mathit{Link}(i,j,k) = \bigcup_{i \le x < j} \{\mathit{Seq}(\mathit{Push}(x,(i,j,k))) \times \mathit{Seq}(\mathit{Cover}(x+1,(i,j,k))) \times \{(i,j,k)\}\}. \tag{13}$$

Here, the symbol × indicates the direct product of sets. That $\mathit{Seq}(\mathit{Push}(x,(i,j,k))) \equiv \mathit{Seq}(x,i,j)$ and $\mathit{Seq}(\mathit{Cover}(x+1,(i,j,k))) \equiv \mathit{Seq}(x+1,j,k)$ follows from their definitions.

Moreover, the relation shown in Figure 8 can be expressed as follows:

$$\mathit{Seq}(i,j,k) = \bigcup_{i \le x < j} \{\mathit{Seq}(\mathit{Insert}(x,(i,j,k))) \times \mathit{Link}(\mathit{Cover}(x,(i,j,k)))\}. \tag{14}$$

Here, that $\mathit{Seq}(\mathit{Insert}(x,(i,j,k))) \equiv \mathit{Seq}(i,x,j)$ and $\mathit{Link}(\mathit{Cover}(x,(i,j,k))) \equiv \mathit{Link}(x,j,k)$ follows from the definitions.

Then, we can provide the formal definition of the $\mathit{Link}(\mathbf{d})$ and $\mathit{Seq}(\mathbf{d})$ for an arbitrary order $\mathbf{d}$ by Exp. (15) and Exp. (17) below.

$$\mathit{Link}(\mathbf{d}) = \bigcup_{\substack{\text{if } d_1=l+1, \text{ then } i=d_1-1; \\ \text{else } i\in[\min(d_0,d_1),\max(d_0,d_1)-1]}} \{\mathit{Seq}(\mathit{Left}(i,\mathbf{d})) \times \mathit{Seq}(\mathit{Right}(i+1,\mathbf{d})) \times \{\mathbf{d}\}\} \tag{15}$$

$$\mathit{where}\ (\mathit{Left}, \mathit{Right}) = \begin{cases} (\mathit{Push}, \mathit{Cover}), \text{ if } d_0 < d_1 \\ (\mathit{Cover}, \mathit{Push}), \text{ if } d_0 > d_1 \end{cases} \tag{16}$$

$$\mathit{Seq}(\mathbf{d}) = \bigcup_{\substack{i\in[\min(d_0,d_1),\ \max(d_0,d_1)] \\ \text{and } i\neq d_1}} \{\mathit{Seq}(\mathit{Insert}(i,\mathbf{d})) \times \mathit{Link}(\mathit{Cover}(i,\mathbf{d}))\} \tag{17}$$

Exp. (15) shows that a complete-link set is recursively composed of the direct product of all possible complete-sequence set pairs, with the N-tuple $\mathbf{d}$ itself.[11] Exp. (17) shows that a complete-sequence set is recursively composed of the direct product of all possible pairs of a complete-link set and a smaller complete-sequence set.

In Exp. (15) and Exp. (17), if $d_0 = d_1$, which violates the restriction of Exp. (8), we then replace d_0 by $\langle /L \rangle$ and $\langle /R \rangle$ as follows. In fact, in this situation alone, $\langle /L \rangle$ and $\langle /R \rangle$ can appear in a $\mathbf{d}$ as d_0, which is mentioned in the definition of Exp. (7) and related properties. The complete sets containing $\langle /O \rangle$ ($O \in \{L, R\}$) tags start the recursive definition.

$$\mathit{Left}(x,\mathbf{d}) = \mathit{Left}(\langle /R \rangle,\mathbf{d}), \qquad \text{if } x = \min(d_0,d_1) \text{ in Exp. (15)} \tag{18}$$

$$\mathit{Right}(x,\mathbf{d}) = \mathit{Right}\ (\langle /L \rangle,\mathbf{d}), \qquad \text{if } x = \max(d_0,d_1) \text{ in Exp. (15)} \tag{19}$$

$$\mathit{Insert}(x,\mathbf{d}) = \mathit{Push}(\langle /L \rangle,\mathbf{d}), \qquad \text{if } x = d_0, \text{ and } d_0 < d_1 \text{ in Exp. (17)} \tag{20}$$

$$\mathit{Insert}(x,\mathbf{d}) = \mathit{Push}(\langle /R \rangle,\mathbf{d}), \qquad \text{if } x = d_0, \text{ and } d_0 > d_1 \text{ in Exp. (17)} \tag{21}$$

4.3 Estimation

According to the recursive definitions, it is natural to derive an inside-outside algorithm (Lari and Young 1990), which is an adaption of the EM algorithm (Dempster, Laird, and Rubin 1977) to tree structures, to conduct parameter re-estimation by calculating the inside and outside probabilities of all complete sets in sentences.

We generalize the expressions in Exp. (15) and Exp. (17) to Exp. (22) and Exp. (23), respectively. In Exp. (22), *Sub$_1$* and *Sub$_2$* mean the $\mathit{Seq}(\mathit{Left}(\cdot))$ and the $\mathit{Seq}(\mathit{Right}(\cdot))$, respectively, on the right-hand side of Exp. (15). In Exp. (23), *Sub$_1$* and *Sub$_2$* mean the $\mathit{Seq}(\mathit{Insert}(\cdot))$ and the $\mathit{Link}(\mathit{Cover}(\cdot))$, respectively, on the right-hand side of Exp. (17). The notation $\langle \cdot, \cdot \rangle$ in Exp. (22) and Exp. (23) represents an unordered two-tuple of a complete-set pair.

[11] We further restrict the *root* mark $\langle /s \rangle$ to take only one modifier (the situation when $d_1 = l + 1$ in Exp. (15)), according to the general restrictions of the dependency grammar.

$$\mathit{Link}(\mathbf{d}) = \bigcup_{\forall \langle \mathit{Sub}_1, \mathit{Sub}_2 \rangle} \{\mathit{Sub}_1 \times \mathit{Sub}_2 \times \{\mathbf{d}\}\} \tag{22}$$

$$\mathit{Seq}(\mathbf{d}) = \bigcup_{\forall \langle \mathit{Sub}_1, \mathit{Sub}_2 \rangle} \{\mathit{Sub}_1 \times \mathit{Sub}_2\} \tag{23}$$

We further define $R_{\text{Link}}(\mathit{Link}(\mathbf{d}), \langle \mathit{Sub}_1, \mathit{Sub}_2 \rangle)$ as a relation for $\mathit{Link}(\mathbf{d})$, $\langle \mathit{Sub}_1, \mathit{Sub}_2 \rangle$ satisfying Exp. (22). Similarly, $R_{\text{Seq}}(\mathit{Seq}(\mathbf{d}), \langle \mathit{Sub}_1, \mathit{Sub}_2 \rangle)$ is a relation for $\mathit{Seq}(\mathbf{d})$, $\langle \mathit{Sub}_1, \mathit{Sub}_2 \rangle$ satisfying Exp. (23). Then, the inside probability β and outside probability α of the two types of complete set can be calculated by Exp. (24) to Exp. (27), where $p(\mathbf{d})$ is the probability of the lexical dependency N-gram, represented by $\mathbf{d}$ in a sentence.

$$\beta(\mathit{Link}(\mathbf{d})) = \sum_{\substack{\langle \mathit{Sub}_1, \mathit{Sub}_2 \rangle,\ \text{s.t.} \\ R_{\text{Link}}(\mathit{Link}(\mathbf{d}), \langle \mathit{Sub}_1, \mathit{Sub}_2 \rangle)}} \beta(\mathit{Sub}_1) \cdot \beta(\mathit{Sub}_2) \cdot p(\mathbf{d}) \tag{24}$$

$$\beta(\mathit{Seq}(\mathbf{d})) = \sum_{\substack{\langle \mathit{Sub}_1, \mathit{Sub}_2 \rangle,\ \text{s.t.} \\ R_{\text{Seq}}(\mathit{Seq}(\mathbf{d}), \langle \mathit{Sub}_1, \mathit{Sub}_2 \rangle)}} \beta(\mathit{Sub}_1) \cdot \beta(\mathit{Sub}_2) \tag{25}$$

$$\alpha(\mathit{Link}(\mathbf{d})) = \sum_{\substack{\langle \mathit{Sup}, \mathit{Con} \rangle,\ \text{s.t.} \\ R_{\text{Seq}}(\mathit{Sup}, \langle \mathit{Link}(\mathbf{d}), \mathit{Con} \rangle)}} \alpha(\mathit{Sup}) \cdot \beta(\mathit{Con}) \tag{26}$$

$$\alpha(\mathit{Seq}(\mathbf{d})) = \sum_{\substack{\langle \mathit{Sup}, \mathit{Con} \rangle,\ \text{s.t.} \\ R_{\text{Link}}(\mathit{Sup}, \langle \mathit{Seq}(\mathbf{d}), \mathit{Con} \rangle)}} \alpha(\mathit{Sup}) \cdot \beta(\mathit{Con}) \cdot p(\mathbf{d}') + \sum_{\substack{\langle \mathit{Sup}, \mathit{Con} \rangle,\ \text{s.t.} \\ R_{\text{Seq}}(\mathit{Sup}, \langle \mathit{Seq}(\mathbf{d}), \mathit{Con} \rangle)}} \alpha(\mathit{Sup}) \cdot \beta(\mathit{Con}) \tag{27}$$

(where $\mathbf{d}'$ is the N-tuple of Sup)

Specifically, Exp. (24) and Exp. (25) can be directly derived from the respective definitions of Exp. (15) and Exp. (17). Further, a complete-link set can only be a component of a complete-sequence set from Exp. (17), while a complete-sequence set can be both a component of a complete-link set from Exp. (15) and a component of a complete-sequence set from Exp. (17). Consequently, Exp. (26) and Exp. (27) can both be derived.

For all $\mathit{Seq}(\mathbf{d})$ with $\langle /L \rangle$ or $\langle /R \rangle$, we use

$$\beta(\mathit{Seq}(\mathbf{d})) = p(\mathbf{d}) \tag{28}$$

as the start of the calculation. At the end of the calculation, the probability of the entire sentence $S = w_0^{l+1}$ can be obtained as follows:

$$P(S) = \beta(\mathit{Seq}(\mathbf{d} = (1, l+1, 0, \cdots, 0))) \tag{29}$$

For re-estimation, we can obtain the probabilistic counts[12] of a dependency N-gram represented by **d** in a sentence using:

$$\frac{\beta(Link(\mathbf{d})) \cdot \alpha(Link(\mathbf{d}))}{P(S)} \quad ^{13} \tag{30}$$

according to the inside-outside algorithm. Finally, all the counts of the dependency N-gram in the training corpus are added and normalized using Exp. (1) to update the model parameters.

We show the details of the re-estimation algorithm with pseudocode in **Appendix A**.

5 Experiments

5.1 Experiment Setting

As the proposed dependency N-gram model and estimation algorithm are independent from language, we conduct experiments using four different languages, i.e., English, German, Spanish and Japanese. The corpora we use for English, German and Spanish are sets of sentences with 5–15 words from the corresponding single-language corpora of **Europarl**[14] (Koehn 2005). The corpus for Japanese is a set of sentences with 5–20 words from the Japanese side of the **NTCIR-8** corpus (Fujii, Utiyama, Yamamoto, Utsuro, Ehara, Echizen-ya, and Shimohata 2010). We take $\frac{1}{200}$ of the sentences from a corpus to form each of the development and test sets used in experiments, and the remaining sentences are used for training. The details of training, development (denoted dev.) and test sets are shown in Table 1 and 2.

To investigate the fundamental properties of the model and algorithm, we do not use any pruning or approximating methods in the parameter estimation. Specifically, we collect all possible lexical dependency N-grams[15] from the raw corpora without any cut-off thresholds for models

Table 1 Training sets

language	sentences	types	tokens
English	400,100	40,913	4,355,333
German	422,951	105,303	4,545,263
Spanish	370,791	58,314	4,007,816
Japanese	477,118	47,930	7,758,437

Table 2 Sentences in dev. and test sets

language	dev. set	test set
English	2,020	2,021
German	2,136	2,136
Spanish	1,872	1,873
Japanese	2,409	2,410

[12] They are no longer integers.
[13] For the situation in Exp. (28), we use $\frac{\beta(Seq(\mathbf{d}))\cdot\alpha(Seq(\mathbf{d}))}{P(S)}$.
[14] http://www.statmt.org/europarl/
[15] As Japanese is a typical head-final language, that is, the head word always comes after its modifiers, we only take the left-oriented (from head to modifier) dependency links into account. For the other three languages, dependency links of both orientations are considered. The parameter collection and initialization do not take the structure into account.

of any order. Before estimation, we use relative frequency to initialize the probabilities.

5.2 Results

5.2.1 Algorithm Convergence

Figure 9 shows the change of English training set perplexities before each iteration by the proposed estimation algorithm, for 2 (bi-) and 3 (tri-) order dependency N-gram models. The convergence trend along with the iteration times can be observed. For the dependency bi-gram, the training set perplexity becomes nearly stable after five iterations. However, for the dependency tri-gram, the first iteration is at very low training set perplexity, and it does not change much in further iterations. This phenomenon suggests that the non-pruned dependency tri-gram model may be complex with many parameters, so the features of the training set are represented well, resulting in a low perplexity. This suggests the model is overfitting the data. We discuss this in Section 5.3.

5.2.2 Test Set Perplexity

As well as the training set perplexity, the perplexity of a test set which has not been used in parameter estimation should be investigated in evaluation. Because different order dependency N-gram models are trained separately, we use linear interpolation in calculating the test set perplexity. Specifically, we use the held-out development set to tune the interpolation coefficients (weights) and to select the iteration times of different order models to minimize the development set perplexity. Next we use the tuned weights to combine the iteration-time-selected models in the test set perplexity calculation. The reason for using simple and straightforward linear interpolation is that we want to discover the essential aspects of the proposed model and algorithm, so we use no further smoothing approaches. As the lowest order of a dependency N-gram is two, we use a uni-gram model with modified KN discounting to handle the unknown words. The uni-gram model is interpolated with the dependency bi-gram model. Furthermore, as the $\langle /L \rangle$

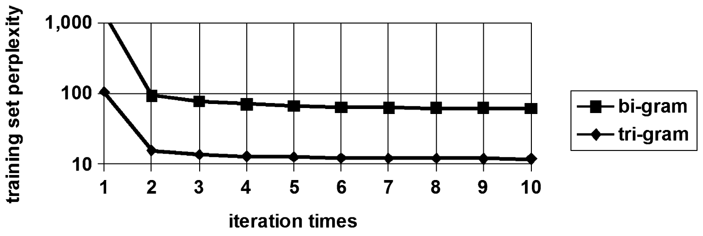

Fig. 9 English training set perplexities before each iteration (y-axis is logarithmic)

and $\langle /R \rangle$ tags are taken as general words, which never really appear in a training set, we treat them separately, and interpolate them with the uni-gram model. Because the $\langle s \rangle$, $\langle L \rangle$ and $\langle R \rangle$ tags only appear in the history, no other model is needed to handle them.

In Table 3, we show the development and test set perplexities of the linear-interpolated dependency bi- and tri-gram models. For comparison, we used **SRILM**[16] (Stolcke 2002) to build two original N-gram language models on the same training sets: one is constructed by maximum likelihood estimation without any smoothing, and the other is constructed by the state-of-the-art interpolated modified KN discounting. We calculate the test set perplexities of the two N-gram language models on the same test sets. The results are listed in Table 4. In both Table 3 and 4, the perplexities are calculated according to the number of lexical words, and the tags used for normalization are not counted[17]. We discuss these results in Section 5.3.

Table 3 Development set perplexities (dev-ppl) and test set perplexities (test-ppl) of dependency N-gram models (N = 2 (bi), 3 (tri))

language	dev-ppl (bi / tri)	test-ppl (bi / tri)	$iter_{bi}$	$iter_{tri}$	λ_{uni}	λ_{bi}	λ_{tri}
English	145 / 143	159 / 156	6	1	0.93	0.99	0.13
German	268 / 256	265 / 261	12	1	0.88	0.98	0.04
Spanish	165 / 164	159 / 158	7	1	0.92	0.99	0.04
Japanese (left-only)	88 / 67	88 / 67	4	1	0.86	0.99	0.70

The iteration times in dependency bi- and tri-gram model training are $iter_{bi}$ and $iter_{tri}$, respectively. The weights of uni-gram, dependency bi- and tri-gram models are λ_{uni}, λ_{bi} and λ_{tri}, respectively. $(1 - \lambda_{bi})$ and $(1 - \lambda_{tri})$ are assigned to the interpolated lower order models and $(1 - \lambda_{uni})$ is assigned to the $\langle /L \rangle$ and $\langle /R \rangle$ tags.

Table 4 Test set perplexities of the original N-gram models

language	MLE (bi / tri)	MKN (bi / tri)
English	162 / 457	157 / 86
German	396 / 1371	252 / 139
Spanish	176 / 499	161 / 86
Japanese	62 / 87	91 / 39

MLE is the maximum likelihood estimation, realized by setting the *adding delta* to 0 in adding smoothing. MKN is the interpolated modified KN discounting.

[16] http://www.speech.sri.com/projects/srilm/

[17] That is, we do not count the $\langle /s \rangle$ tag in the original N-gram language models, or $\langle /L \rangle$ and $\langle /R \rangle$, in our models. If they are included, the perplexities decrease. In the original N-gram model, this is because a $\langle /s \rangle$ tag nearly always appears after the period mark. The effect is even more dramatic in our model, as each word in a sentence has a $\langle /L \rangle$ and a $\langle /R \rangle$ tag to normalize its modifier numbers, so the token number in a sentence is multiplied. Therefore, for fairness, we only count the lexical words in the perplexity calculation.

5.3 Discussion

5.3.1 Parameter Number

For a sentence with l words, the number of dependency N-grams that can be collected increases exponentially as $O(l^N)$ if we consider all possible combinations. Although for a given N, the proposed algorithm takes a time which is polynomial in sentence length l, a large N will be practically intractable, especially for long sentences. In Figure 10, we show the number of complete sets of different order dependency N-gram models for different sentence lengths.

This behavior is also related to the overfitting problem, because our algorithm is essentially an iterative maximum likelihood estimation. A model that is very complex will be extremely specific to the training set. From Table 3, we see that the performance of a dependency tri-gram model will saturate after only one iteration, which is also indicated in Figure 9, and does little to improve the test set perplexities. The exception is Japanese, where the dependency tri-gram does improve the performance. The linguistic reason for this is that Japanese is a head-final language with a simpler syntactic structure, so we restrict the dependency link in Japanese to "left only", which leads to a model with fewer parameters. Consequently, the high order model performs better. From the experimental results, we can see that the proposed algorithm has the usual strengths and weaknesses of an EM algorithm.

5.3.2 Test Set Perplexity

Comparing the test set perplexities in Table 3 and 4, we can see the dependency bi-gram model achieves the same, or sometimes better, performance as the original N-gram language models. However, when we look at the tri-grams, the interpolated modified KN discounting method, which is the state-of-the-art, shows its strength, and our dependency model does not produce much improvement for the reasons we described above[18]. As the modified KN method

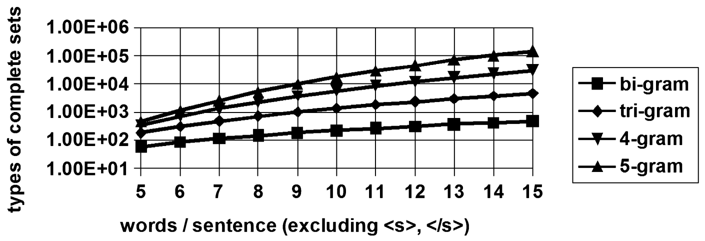

Fig. 10 Number of complete sets (y-axis is logarithmic)

[18] For Japanese, the result is improved by the dependency tri-gram model, but the original tri-gram model with interpolated modified KN discounting method performs much better.

uses an efficient discounting to avoid the overfitting problem and our model has no smoothing, the difference in performance is reasonable for complex models. Conversely, the competitive results of our bi-gram model and its performance on Japanese show that our model is a promising one, particularly if the number of parameters can be reduced.

5.3.3 Model Preference

In Figure 11 to 23, we present the examples of the best dependency structures of sentences in test set generated by our approach. We used the settings in Table 3 and generated them using the Viterbi algorithm (Viterbi 1967). It can be seen that the proposed approach can reveal features of specific languages, even though it is unsupervised, such as for the final-position verb "*stellen*" and its relation with the second-position auxiliary verb "*möchte*" in the German sentence in Figure 15. The results also show a preference for associating semantic relations and making the function words[19] of a language the modifiers of the content words. This tendency is noticeable in the English examples, such as the particle "*to*" in Figure 12 and 14 and the "*'s*" in Figure 14. In addition, the arrangement of commas around "*however*" in Figure 11 and around "*therefore*" in Figure 14 is impressive. Another example is in the Spanish sentence in Figure 19. Syntactically, the preposition "*a*" is the head of the noun "*respecto*", but in unsupervised training, our model prefers to assign "*a*" to be the modifier of "*respecto*" and directly link two content words, i.e. "*respecto*" and the verb "*haciendo*". We think this is because the probabilities of $\langle /L \rangle$ and $\langle /R \rangle$ tags have large estimates, especially when they appear after function words, which prevents them from having modifiers. This tendency, however, is correct for articles, such as the "*der*" in German and "*la*" in Spanish. Furthermore, an interesting phenomenon can be observed in the Japanese sentence in Figure 23. In the example, the verb stem "応用" is linked to the auxiliary verb "できる", and the words of "する こと が" are arranged as a dependency chain and attached to "できる" as well. Semantically, the expressions of "応用 できる" (literally: *can apply*) and "応用 する こと が できる" (literally: *the thing, that applies, can*) have nearly the same meaning and are generally interchangeable. Consequently, the model prefers to designate "する こと が", which has a weak semantic function, as a branch and link the semantically-crucial words "応用" and "できる" directly. All these examples suggest that the proposed model with unsupervised training has a strong preference for organizing a sentence by semantic relations and for assigning relations between those words that play a central role in such a semantic unit.

[19] Articles, prepositions, etc.

Example of English Sentences

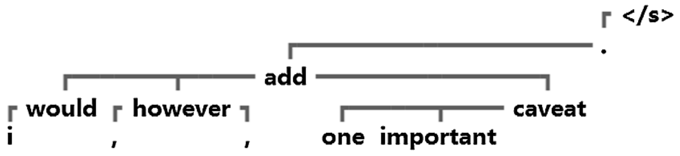

Fig. 11 Best dependency structure of the sentence “*i would , however , add one important caveat .*”

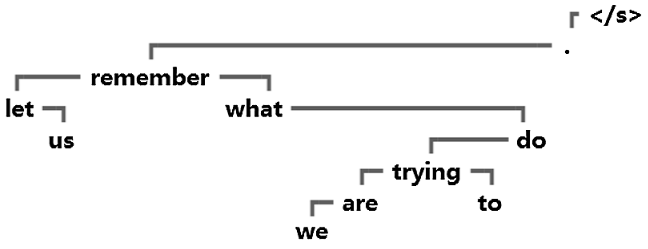

Fig. 12 Best dependency structure of the sentence “*let us remember what we are trying to do .*”

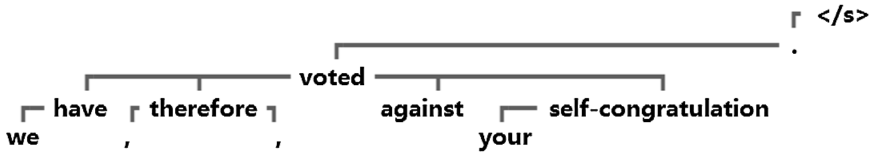

Fig. 13 Best dependency structure of the sentence “*we have , therefore , voted against your self-congratulation .*”

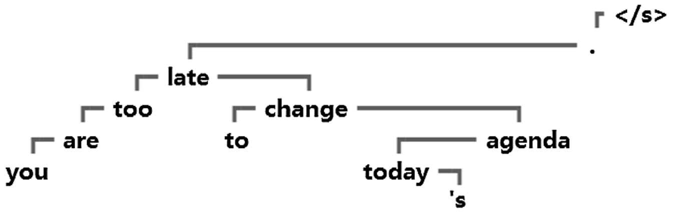

Fig. 14 Best dependency structure of the sentence “*you are too late to change today 's agenda .*”

Example of German Sentences

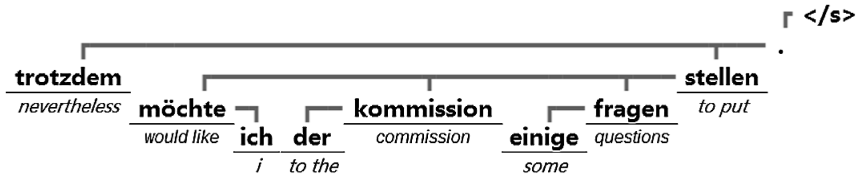

Fig. 15 Best dependency structure of the sentence “*trotzdem möchte ich der kommission einige fragen stellen .*”

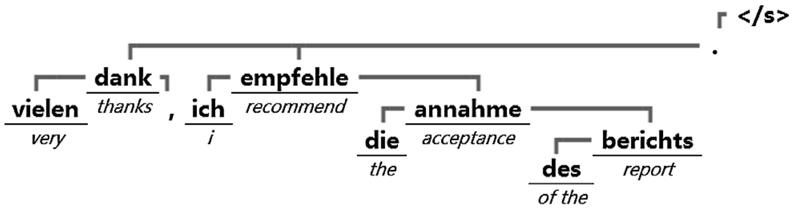

Fig. 16 Best dependency structure of the sentence “*vielen dank , ich emphehle die annahmen des bericht .*”

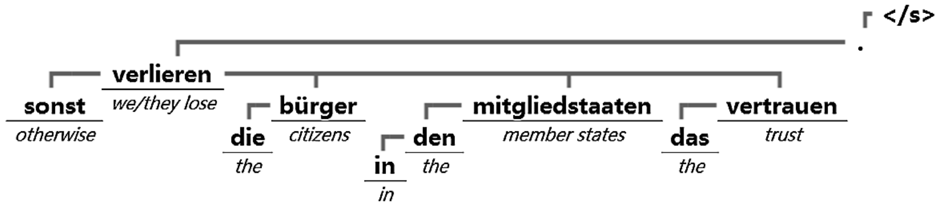

Fig. 17 Best dependency structure of the sentence “*sonst verlieren die bürger in den mitgliedstaaten das vertauen .*”

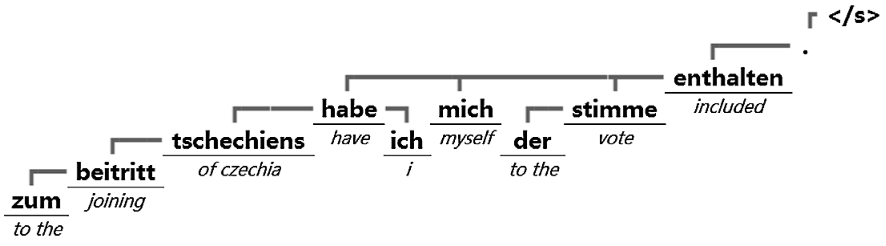

Fig. 18 Best dependency structure of the sentence “*zum beitritt techechiens habe ich mich der stimme enthalten .*”

Example of Spanish Sentences

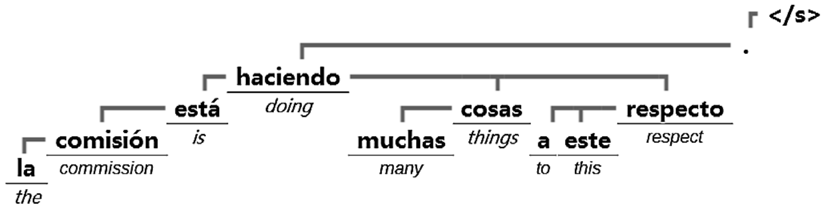

Fig. 19 Best dependency structure of the sentence "*la comisión está haciendo muchas cosas a este respecto .*"

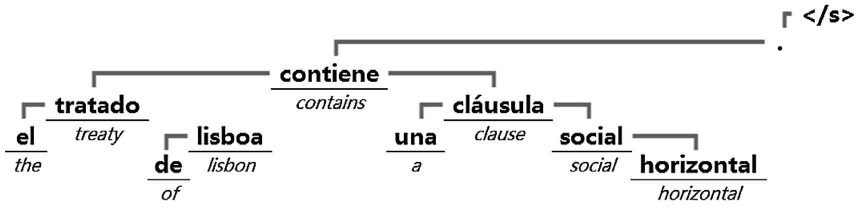

Fig. 20 Best dependency structure of the sentence "*el tratado de lisboa contiene una cláusula social horizontal .*"

Example of Japanese Sentences

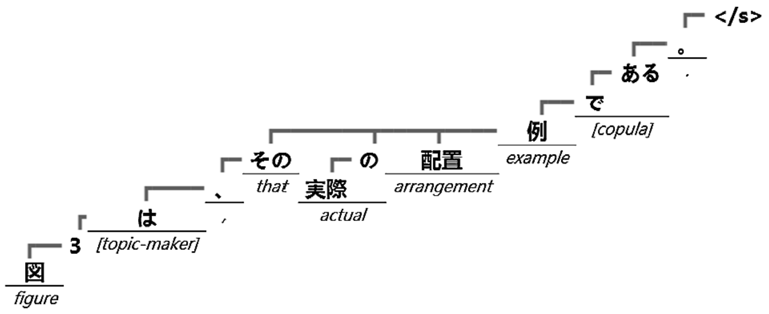

Fig. 21 Best dependency structure of the sentence"図 3 は 、 その 実際 の 配置 例 で ある 。"

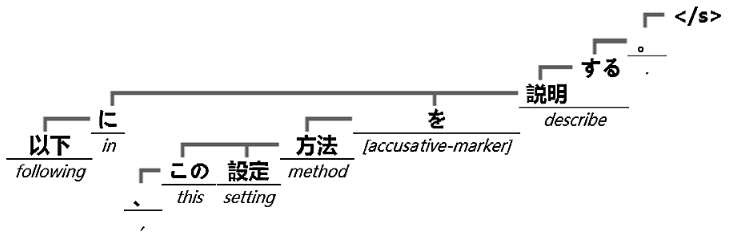

Fig. 22 Best dependency structure of the sentence"以下 に 、 この 設定 方法 を 説明 する 。"

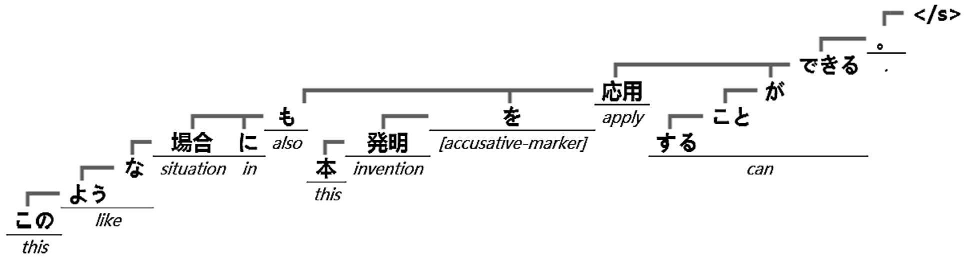

Fig. 23 Best dependency structure of the sentence "この よう な 場合 に も 本発明 を 応用 する こと が できる 。"

6 Application to Microblog Data

6.1 Task and Corpus

Microblogging is an emerging application that provides a new platform for communicating. Postings on microblogs are usually brief due to restrictions on message length such that no more than 140 characters may be used. Moreover, microblogs use many non-standard expressions and Internet-based neologisms. These words and expressions are hard for general natural language processing tools, which are usually trained on standard data sets, to handle. Therefore, tasks using microblogs as a huge data source must consider the characteristics of user-generated content. One example of this is the part-of-speech tagging task on microblogs (Gimpel, Schneider, O'Connor, Das, Mills, Eisenstein, Heilman, Yogatama, Flanigan, and Smith 2010). However, the more explicit and structured we want the information extracted from a microblog to be, the more difficult the task turns out to be, due to the flexible and non-standard use of the expressions.

Because our proposed approach is completely data driven, we think it can efficiently capture features in user-generated content. In this section, using our proposed model, we focus on extracting and investigating lexical dependency features from Chinese microblog data.

We use the NLPIR Chinese Weibo corpus[20] in our experiment. The corpus contains 230,000 posts collected from *Sina Weibo* and *Tencent Weibo*. During preprocessing, we split the posts into sentences, delete the tags beginning with @ and # and all the URLs. We use the Stanford Chinese Word Segmenter[21] (Tseng, Chang, Andrew, Jurafsky, and Manning 2005) with the *Chinese Penn Treebank standard* to segment each Chinese sentence and take those sentences with 5–30 words as the training corpus in our experiment. We further normalize the punctuation marks in the

[20] http://www.nlpir.org/download/weibo_content_corpus.rar
[21] http://www-nlp.stanford.edu/software/segmenter.shtml

Table 5 Chinese Weibo corpus

sentences	types	tokens
304,667	95,239	4,427,543

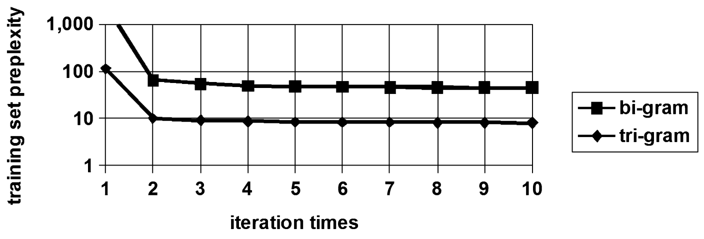

Fig. 24 Training set perplexities before each iteration (y-axis is logarithmic)

training corpus and replace all words appearing fewer than five times with a symbolic *UNK* token. The details of the training corpus are shown in Table 5.

We collect all possible lexical dependency N-grams from the training corpus and use relative frequency to initialize the probabilities. Figure 24 shows the change of training corpus perplexities before each iteration for 2 (bi-) and 3 (tri-) order dependency N-gram models. We can observe the same trend shown in Figure 9. However, our interest is the learned features of the training set. We discuss the examples of lexical parameters and parsing using a three-time iterated dependency bi-gram model and a one-time iterated dependency tri-gram model in the next section.

6.2 Discussion

In Table 6, 7 and 8, we show the examples of lexical dependency features with high estimates of logarithmic probability (log-prob.) with our unsupervised approach.

In Table 6, the examples of dependency bi-grams around general words are shown. We can see that the dependency relations between them are well modeled. Moreover, in Table 6, we show the features of the *root* mark $\langle /s \rangle$ of sentences. We can see that the final punctuation marks and final-position particles are automatically learned as modifiers of the root mark (i.e., of the root word of a sentence). In Table 7, we show an example of a special Chinese expression on the Internet. The word "神马", read as *shén-mǎ*, means "*what*" on the basis of similarity in pronunciation to the original word *shén-me*. Our unsupervised data-driven approach reveals the behaviors of this neologism, which is natural for a Chinese native speaker.

In Table 8, we show an example of a dependency tri-gram. The expression "有木有" (*yǒu-mù-yǒu*) is also an Internet-based neologism, which means "*to exist or not*", because the original

Table 6 Dependency bi-grams for the *root* mark ($\langle /s \rangle$) and some general Chinese words

bi-gram	log-prob.	bi-gram	log-prob.	bi-gram	log-prob.	bi-gram	log-prob.
</s> .	−0.520	是 不 [copula] not	−1.358	是 [copula] 个 [counter]	−2.278	做 do 的 [particle]	−1.371
</s> !	−1.098	是 就 [copula] just	−1.628	是 [copula] 不 not	−2.755	做 do 了 already	−1.436
</s> ?	−1.235	是 这 [copula] this	−1.915	是 [copula] 一 one	−2.892	做 do 个 [counter]	−1.925
</s> 吗 [question-maker]	−1.928	是 都 [copula] all	−2.139	是 [copula] 我 i	−3.058	做 do 什么 what	−2.085
</s> [	−1.955	是 也 [copula] also	−2.261	做 do 的 [particle]	−3.262	做 do 得 [particle]	−2.102

The top five highest estimates for each history (given o^1 and h^1 in (h^0, o^1, h^1)) are shown. The h^0 shown here excludes *UNK*, $\langle /L \rangle$, $\langle /R \rangle$ and punctuation marks for lexical h^0.

Table 7 Dependency bi-grams for a Chinese Internet neologism “神马”

bi-gram	log-prob.	bi-gram	log-prob.
神马 是 *what [copula]	−1.327	神马 *what 的 [particle]	−0.972
神马 这 *what this	−1.343	神马 *what 都 all	−1.343
神马 为 *what for	−1.552	神马 *what 啊 ah	−1.532
神马 城管 *what chengguan	−1.741	神马 *what 情况 situation	−1.803
神马 你 *what you	−1.813	神马 *what 时候 time	−1.832

In (h^0, o^1, h^1), h^1 = 神马, $o^1 = \langle L \rangle$ or $\langle R \rangle$. The top five highest estimates are shown.

Table 8 Dependency tri-grams for a Chinese Internet neologism “有木有” and for the corresponding standard expression “有没有”

bi-gram	log-prob.	bi-gram	log-prob.
有 没 有	−0.713	有 木 有	−0.537
有 有 没	−0.736	有 木 有	−0.732
有 有 没	−0.840	有 有 木	−1.555
有 没 有	−0.957	有 有 木	−1.746

The estimates of all four possible structure patterns of both expressions are shown.

expression “有没有” (*yǒu-méi-yǒu*) has a similar pronunciation in some dialects. We show all four possible structure patterns for both of them. We can see that all of these structures have relatively high estimates, which shows strong dependency relations. However, if we use a general Chinese parser[22], the character “木” is always treated as a noun due to its original meaning of “*wood*”, and the tri-order relation of the expression “有木有” is not recognized.

In Figure 25, we show the best dependency structure given by the Viterbi algorithm for a

[22] http://nlp.stanford.edu:8080/parser/index.jsp

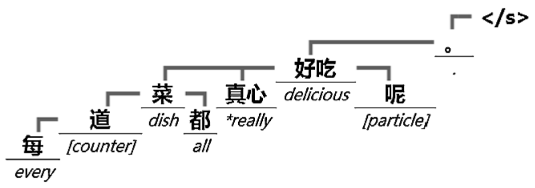

Fig. 25 Best dependency structure of the sentence "每 道 菜 都 真心 好吃 呢。" generated by the proposed approach

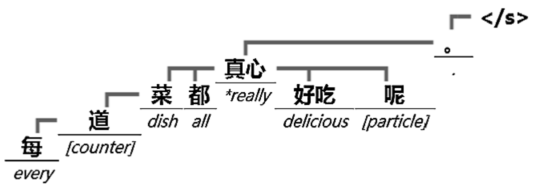

Fig. 26 Dependency structure of the sentence "每 道 菜 都 真心 好吃 呢。" generated by the Stanford Dependency Parser

sentence in the training set. Figure 26 is the dependency structure generated by the Stanford Dependency Parser[6.2]. Although there are no neologisms in this sentence, the Chinese word "真心" is a general word and ordinarily used as a noun or an adjective, meaning "*sincerity*" or "*sincere*" respectively. However, it has recently been used as an adverb to express the meaning of "*really*" in a slightly emphatic manner. This feature is also captured by our approach from training data and contextualized within the entire sentence structure, resulting in a correct analysis. A standard parser cannot efficiently handle this type of flexible usage of general words.

7 Conclusion and Future Work

In this study, we focused on introducing sentence structure into language modeling. We proposed a generative dependency N-gram language model, which extends the original N-gram language model to include sentence dependency structures, as well as a definition of complete sets for arbitrary order, which facilitates an unsupervised parameter estimation algorithm. The experimental results demonstrate the applicability and the properties of the proposed approach. We also applied a complete data-driven approach for lexical dependency feature extraction to a textual microblog data set. The experimental results show that our approach can handle non-standard linguistic phenomena in user-generated content.

In future, we will develop methods for parameter pruning and discounting to handle the overfitting problem. As the proposed dependency language model is intrinsically complex, we also plan more fundamental simplifications. In addition, although our proposed algorithm is unsupervised, the output of a trained parser, which would provide clear and lexical heuristics, can be integrated in it. We plan to investigate this possibility and evaluate the performance achieved by using linguistically motivated criteria.

Acknowledgement

An earlier version of this study has been presented at *the 6th International Joint Conference on Natural Language Processing (IJCNLP 2013)* (Ding and Yamamoto 2013). We would like to thank the anonymous reviewers for their valuable comments and suggestions. This study was supported by JSPS KAKENHI Grant Number 24650063.

Reference

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). "Class-based N-gram Models of Natural Language." *Computational Linguistics*, **18** (4), pp. 467–479.

Chelba, C. and Jelinek, F. (2000). "Structured Language Modeling." *Computer Speech and Language*, **14** (4), pp. 283–332.

Chen, S. F. and Goodman, J. (1998). "An Empirical Study of Smoothing Techniques for Language Modeling." Tech. rep., TR-10-98, Computer Science Group, Harvard Univ.

Collins, M. (1998). "Three Generative, Lexicalised Models for Statistical Parsing." In *Proceedings of ACL 1998*, pp. 16–23.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society. Series B (Methodological)*, **39** (1), pp. 1–38.

Ding, C. and Yamamoto, M. (2013). "An Unsupervised Parameter Estimation Algorithm for a Generative Dependency N-gram Language Model." In *Proceedings of IJCNLP2013*, pp. 516–524.

Eisner, J. M. (1996). "Three New Probabilistic Models for Dependency Parsing: An Exploration." In *Proceedings of COLING1996*, pp. 340–345.

Fujii, A., Utiyama, M., Yamamoto, M., Utsuro, T., Ehara, T., Echizen-ya, H., and Shimohata, S. (2010). "Overview of the Patent Translation Task at the NTCIR-8 Workshop." In *Proceedings of NTCIR-8*, pp. 371–376.

Gao, J. and Suzuki, H. (2003). "Unsupervised Learning of Dependency Structure for Language Modeling." In *Proceedings of ACL2003*, pp. 521–580.

Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., and Smith, N. A. (2010). "Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments." Tech. rep., DTIC Document.

Graham, Y. and van Genabith, J. (2010). "Deep Syntax Language Models and Statistical Machine Translation." In *Proceedings of SSST at COLING2010*, pp. 118–126.

Klein, D. and Manning, C. D. (2003). "Accurate Unlexicalized Parsing." In *Proceedings of ACL2003*, pp. 423–430.

Klein, D. and Manning, C. D. (2004). "Corpus-based Induction of Syntactic Structure: Models of Dependency and Constituency." In *Proceedings of ACL2004*, pp. 478–485.

Koehn, P. (2005). "Europarl: A Parallel Corpus for Statistical Machine Translation." In *Proceedings of MT summit 2005*, pp. 79–86.

Koo, T. and Collins, M. (2010). "Efficient Third-order Dependency Parsers." In *Proceedings of ACL2010*, pp. 1–11.

Kudo, T. and Matsumoto, Y. (2002). "Japanese Dependency Analysis using Cascaded Chunking." In *Proceedings of COLING2002*, pp. 1–7.

Lari, K. and Young, S. J. (1990). "The Estimation of Stochastic Context-free Grammars using the Inside-Outside Algorithm." *Computer Speech and Language*, **4** (1), pp. 35–56.

Lee, S. and Choi, K.-S. (1997). "Reestimation and Best-first Parsing Algorithm for Probabilistic Dependency Grammars." In *Proceedings of WVLC1997*, pp. 41–55.

Lee, S. and Choi, K.-S. (1998). "Automatic Acquistion of Language Model Based on Head-dependent Relation between Words." In *Proceedings of COLING-ACL1998*, pp. 723–727.

Nivre, J. (2008). "Algorithms for Deterministic Incremental Dependency Parsing." *Computational Linguistics*, **34** (4), pp. 513–553.

Paskin, M. A. (2002). "Grammatical Bigrams." *Advances In Neural Information Processing Systems 14*, **1**, pp. 91–97.

Stolcke, A. (2002). "SRILM—An Extensible Language Modeling Toolkit." In *Proceedings of ICSLP2002*, pp. 901–904.

Stolcke, A., Chelba, C., Engle, D., Jimenez, V., Mangu, L., Printz, H., Ristad, E., Rosenfeld, R., Wu, D., Jelinek, F., and Khudanpur, S. (1997). "Dependency Language Modeling.".

Tillmann, C. and Ney, H. (1997). "Word Triggers and the EM Algorithm." In *Proceedings of CoNNL 1997*, pp. 117–124.

Tseng, H., Chang, P., Andrew, G., Jurafsky, D., and Manning, C. (2005). "A Conditional Random

Field Word Segmenter." In *SIGHAN2005, Workshop on Chinese Language Processing*, Vol. 171.

Viterbi, A. (1967). "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm." *Information Theory, IEEE Transactions on Information Theory*, **13** (2), pp. 260–269.

Wetherell, C. S. (1980). "Probabilistic Languages: A Review and Some Open Questions." *ACM Computing Surveys (CSUR)*, **12** (4), pp. 361–379.

Zhang, H. and McDonald, R. (2012). "Generalized Higher-order Dependency Parsing with Cube Pruning." In *Proceedings of EMNLP2012*, pp. 320–331.

Appendix

A Calculation of the Estimation Algorithm

In Section 4.3, we provide a re-estimation method as an inside-outside algorithm. When the order is restricted to two, as in (Lee and Choi 1997), the calculation of all the inside ($\beta(\cdot)$) and outside ($\alpha(\cdot)$) probabilities can be conducted on a CKY-wise triangle table. Specifically, the inside probabilities are first calculated from small to large spans in a bottom-up way; then the outside probabilities are calculated in a top-down way by using the calculated inside probabilities. Because the proposed approach extends the dependency N-gram to an arbitrary order, a two-dimensional triangle table is not sufficient. In general, the re-estimation of a model with N-order lexical parameters needs an N-dimension table, which ceases to be intuitive.

A naive method of calculating the inside-outside probability of a sentence can be performed by the pseudocode of Naive-Inside-Outside. More efficiently, we can first generate a *process list* to record the processing order for all the necessary inside and outside probabilities of a sentence, and then calculate them according to the process list. In practice, the process given in Naive-Inside-Outside, which identifies the processing order, needs to execute only once, which can be performed during pre-processing. The generation of the process list is shown by the pseudocode of Process-List. This code is basically identical to the process Naive-Inside-Outside; instead of calculating, the "name" tokens (generated by Get-Tuple) of all the inside and outside probabilities are recorded. For creating the process list, the calculation step of Inside-Outside is trivial, because the calculation is found from the recorded "name" by using Get-Probability.

```
NAIVE-INSIDE-OUTSIDE(SENTENCE)
 1  calculate $\beta(Set)$ according to Exp. (28)
 2  repeat
 3          for all inside probabilities $\beta(Set)$ of SENTENCE
 4              do ▷ according to Exp. (24), Exp. (25)
 5                 if all $\beta(Sub_1)$, $\beta(Sub_2)$ on the right side are available
 6                   then calculate $\beta(Set)$
 7     until all inside probabilities $\beta(Set)$ are calculated
 8  repeat
 9          for all outside probabilities $\alpha(Set)$ of SENTENCE
10              do ▷ according to Exp. (26) or Exp. (27)
11                 if all $\alpha(Sup)$ on the right side are available
12                   then calculate $\alpha(Set)$
13     until all outside probabilities $\alpha(Set)$ are calculated
```

```
INSIDE-OUTSIDE(SENTENCE, PROCESSLIST)
1  for $i \leftarrow 1$ to length (PROCESSLIST)
2       do GET-PROBABILITY(PROCESSLIST[$i$])
```

```
GET-PROBABILITY(X, Y, Z)
1  return X (Y (Z)) according to Exp. (24), Exp. (25), Exp. (26), Exp. (27) or Exp. (28)
```

```
PROCESS-LIST()
 1  Process-List ← []
 2  for all $Seq(\mathbf{d})$ of Exp. (28)
 3       do append GET-TUPLE($\beta(Seq(\mathbf{d}))$) to Process-List
 4  repeat
 5          for all complete sets $Set$
 6              do ▷ according to Exp. (24) or Exp. (25)
 7                 if all GET-TUPLE($\beta(Sub_1)$), GET-TUPLE($\beta(Sub_2)$) $\in$ Process-List
 8                   then append GET-TUPLE($\beta(Set)$) to Process-List
 9     until GET-TUPLE($\beta(Set)$) $\in$ Process-List for all $Set$
10  repeat
11          for all complete sets $Set$
12              do ▷ according to Exp. (26) or Exp. (27)
13                 if all GET-TUPLE($\alpha(Sup)$) $\in$ Process-List
14                   then append GET-TUPLE($\alpha(Set)$) to Process-List
15     until GET-TUPLE($\alpha(Set)$) $\in$ Process-List for all $Set$
16  return Process-List
```

```
GET-TUPLE(X (Y (Z)))
1  return (X, Y, Z)
```

Chenchen Ding: He received a B.S. degree in Mathematics from Shandong University, China, in 2009. He received an M.E. degree in Computer Science from the University of Tsukuba, Japan, in 2012. Currently, he is a doctoral student of the University of Tsukuba, working on natural language processing.

Mikio Yamamoto: He received the B.E, M.E and D.E degrees in information and computer sciences from Toyohashi University of Technology, Japan, in 1984, 1986 and 1992, respectively. In 1988, he joined the Toyohashi University of Technology as a technical official. In 1995, he moved to the University of Tsukuba as an assistant professor. Currently, he is a professor in the Department of Computer Science, University of Tsukuba, working on statistical methods for processing text.

(Received October 29, 2013)
(Revised February 4, 2014)
(Rerevised April 12, 2014)
(Accepted May 22, 2014)