An Unsupervised Parameter Estimation Algorithm for a Generative Dependency N-gram Language Model

Chenchen Ding and **Mikio Yamamoto**
Department of Computer Science
University of Tsukuba
1-1-1 Tennodai, Tsukuba, 305-8573, Japan
{tei@mibel.,myama@}cs.tsukuba.ac.jp

Abstract

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 kinds of extra tags 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: English, German, Spanish and Japanese. The results illustrate the applicability and the properties of the proposed approach.

1 Introduction

Statistical language models are a fundamental component of speech recognition systems, machine translation systems, and so forth. Presently, the N-gram language model is the most widely used approach. This model focuses on sequences of neighboring lexical words (Fig. 1), and uses the probabilities of these sequences as model parameters. Due to the full 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 regards a sentence as a flat string and ignores its structure.

Figure 1: The N-gram language model treats the English sentence "*all things pass*" composed of *(all, things)* and *(things, pass)*, for $N = 2$.

Figure 2: The constituency-based parsing **(A)** and the dependency-based parsing **(B)**[1] for the English sentence "*all things pass*".

On the other hand, revealing the structure of a sentence is the task of parsing, which is based on linguistically oriented formulations and focuses on generating the likeliest structure for a given sentence. For this purpose, there are constituency-based and dependency-based formulations (Fig. 2). The former organizes continuous word sequences in a hierarchy of small range to large range groups with linguistically oriented labels; the latter links words with dependency relations[2].

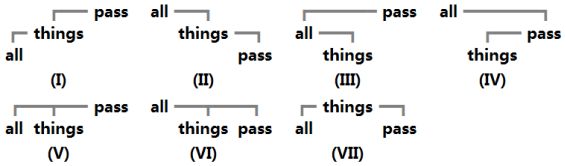

Figure 3: All possible dependency structures for the English sentence "*all things pass*". **(I)** is the linguistically correct structure while the original N-gram language model handles the sentence as if it has the structure labeled **(II)**. We consider all these structures in our unsupervised estimation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

In this paper, we focus on introducing sentence structure into language modeling. We propose a generative dependency N-gram language model that integrates a generative dependency structure of a sentence into the original N-gram language model. We prefer the dependency-based formula-

[1] For the illustrations in this paper, 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.

[2] 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.

International Joint Conference on Natural Language Processing, pages 516–524,
Nagoya, Japan, 14-18 October 2013.

tion 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 fully lexical as the original N-gram language model. We further propose an expectation-maximization (EM) algorithm for estimating the probability of arbitrary order[3] dependency N-grams, by considering all possible dependency structures[4] of a sentence (Fig. 3). 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 discounting proposed in it is still the state-of-the-art approach. 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 et al., 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 basically take a sentence as a flat string.

Many approaches have been proposed for constituency-based parsing (Collins, 1998; Klein and Manning, 2003; Klein and Manning, 2004) and for dependency-based parsing (Eisner, 1996; Lee and Choi, 1997; Kudo and Matsumoto, 2002; Klein and Manning, 2004; Nivre, 2008). Presently, 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) and it is hard to handle more lexical information.

There have been some 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 (Stolcke et al., 1997; Gao and Suzuki, 2003; Graham and van Genabith, 2010). However, these approaches need a trained dependency parser because they construct a language model based on the decisive best structure produced by the parser.

In our approach, we utilize a generative dependency model to guarantee the constituency of a language model[5], but our model and algorithm can handle arbitrary numbers of lexical words. Furthermore, our approach needs no extra parser to generate the best structure of a sentence but, instead, takes all possible dependency structures into consideration.

3 Generative Dependency Model

We model the marginal probability of a sentence S over the 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 fully lexical word sequences with dependency relations. However, the term $P(d)$ is difficult to model and is usually taken to be a constant, as in Paskin (2002). To deal with this problem, the *dependency model with valence* (DMV)[6] proposed by Klein and Manning (2004) introduces a special mark *STOP*. However, it is necessary to distinguish two kinds of parameters, 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 the DMV, we introduce four kinds of tags to normalize the distribution of modifier numbers (the valence) of a head word. In this paper, we use $\langle L \rangle$, $\langle /L \rangle$, $\langle R \rangle$ and $\langle /R \rangle$ to show the start and end of the left and right modifier word sequences of a head word, respectively. The dependency structure can thus be organized as nested word sequences. Specifically, a modifier word sequences of a head word is in a form of $M = m_0^{\phi+1} \equiv m_0, m_1, \cdots, m_{\phi+1}$, 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 Fig. 4. On the other hand, in contrast to the DMV, we treat the tags as ordinary words in the parameter estimation. So the parameters of our model have a uniform representation, by which our approach can be easily extended to arbitrary high orders.

[3] "Order" here means the number of lexical words (N).

[4] Only projective dependency structures are considered.

[5] $\sum_{S \in L} P(S) = 1$ for the set L composed of all the sentences S in a language.

[6] A generative model.

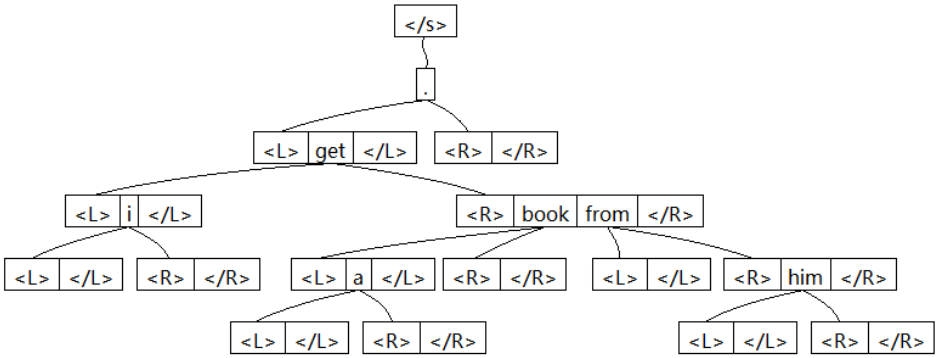

Figure 4: A dependency structure for the English sentence "*i get a book from him .*", with $\langle L\rangle$, $\langle R\rangle$, $\langle /L\rangle$, $\langle /R\rangle$ tags. The root of the sentence is marked as $\langle /s\rangle$ and for a word without modifiers, its modifier word sequences are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

Because our model is essentially equivalent to the generative *Model C* in Eisner (1996), the consistence of the language model can be guaranteed. That is, $\langle O\rangle m_1^{\phi}\langle /O\rangle$ ($O \in \{L, R\}$) 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, so: the larger $P(m_{\phi+1} = \langle /O\rangle)$ is, the smaller ϕ, which is the number of generated words, becomes, and vice versa.

Without loss of generality, the probability of $m_{\kappa+1}$ ($0 \le \kappa \le \phi$) 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[7]. We use the independent assumption that the probability of a word in the generation process depends on only its direct ancestors and the orientation between them. So, the general probability can be simplified to:

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

where h^k is a lexical word, h^{k+1} is the head word of h^k, and $o^k \in \{\langle L\rangle, \langle R\rangle\}$ is retained in the history to show the dependency orientation. $\langle /L\rangle$ and $\langle /R\rangle$ tags can and only can[8] take the place of h^0.

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 in this paper. For example, a dependency N-gram is ($\langle /L\rangle$, $\langle L\rangle$, *him*, $\langle R\rangle$, *from*, $\langle R\rangle$, *get*, $\langle L\rangle$, ., $\langle /s\rangle$) in the dependency structure illustrated in Fig. 4. 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 order parameters.

[7]The generation process can be realized in a depth-first or a breadth-first way but the distinction is unessential.

[8]Because they cannot have further modifiers.

4 Parameter Estimation

4.1 Notation

For a sentence $S = w_0^{l+1} \equiv w_0, w_1, \cdots, w_{l+1}$, where $w_0 \equiv \langle s\rangle$ and $w_{l+1} \equiv \langle /s\rangle$, a dependency N-gram $(h^0, o^1, h^1, \ldots, o^{n-1}, h^{n-1})$ can be denoted by $\mathbf{d} = (d_0, d_1, \ldots, d_{n-1})$ where $h^k = w_{d_k}$. That is, a dependency N-gram can be denoted by an N-tuple of the absolute positions of words in a given sentence. As the magnitudes of d_k and d_{k+1} show the orientation, o^{k+1} can be omitted.[9]

Lee and Choi (1997) propose the *complete-link set* and *complete-sequence set* for head-modifier pair (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, i.e. 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 kinds of sets can be defined recursively and the set of all possible dependency

[9]If h^0 is $\langle /L\rangle$ or $\langle /R\rangle$, we retain them in $\mathbf{d}$. The orientations can also be unambiguously omitted for these two tags.

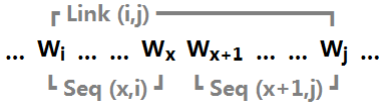

Figure 5: $Link(\mathbf{d} = (i, j))$. In Lee and Choi (1997), for a span $[i, j]$, $Link(i, j)$ is composed of the dependency link of w_i and w_j, and all possible pairs of complete-sequence sets $Seq(x, i)$ and $Seq(x + 1, j)$.

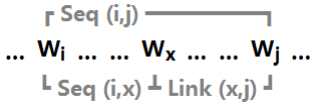

Figure 6: $Seq(\mathbf{d} = (i, j))$. In Lee and Choi (1997), for a span $[i, j]$, $Seq(i, j)$ is composed of all possible pairs of complete-sequence set $Seq(i, x)$ and complete-link set $Link(x, j)$.

structures of a sentence $S = w_0^{l+1}$ is the complete-sequence set over the span $[1, l + 1]$. We illustrate these recursive relations in Fig. 5 and Fig. 6.

Because more than two words are involved in the proposed dependency N-gram, we generalize the two kinds of sets 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 kinds of sets. We show examples of a dependency tri-gram in Fig, 7 and Fig. 8.

4.2 Recursive Definition

Here, we give the formulation of the recursive definition of the complete-link set and complete-sequence set for an arbitrary order dependency N-gram. 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 to guarantee that a head word is outside of the range covered by a chain of its descendants.

$$\begin{aligned} &d_k > \max(d_0, \cdots, d_{k-1}), \text{ or} \\ &d_k < \min(d_0, \cdots, d_{k-1}) \end{aligned} \tag{2}$$

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 the $\langle /s \rangle$. So, we have the following constraints.

$$d_{k+1} = 0, \text{ if } d_k = l + 1, \text{ or } d_k = 0 \tag{3}$$

For convenience, we introduce three kinds of operations, *Push*, *Cover*, and *Insert* over an index x (absolute word position) and an N-tuple $\mathbf{d} = (d_0, d_1, \ldots, d_{n-1})$:

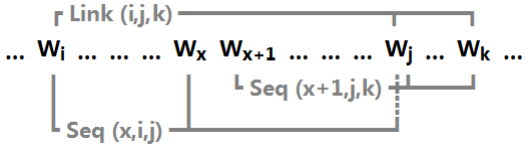

Figure 7: $Link(\mathbf{d} = (i, j, k))$. In our model, an extended high-order (3 is shown here) complete-link set $Link(i, j, k)$ is composed of the N-tuple $\mathbf{d}$, and all possible pairs of complete-sequence sets $Seq(x, i, j)$ and $Seq(x + 1, j, k)$.

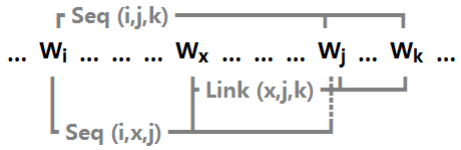

Figure 8: $Seq(\mathbf{d} = (i, j, k))$. In our model, an extended high-order (3 is shown here) complete-sequence set $Seq(i, j, k)$ is composed of all possible pairs of complete-sequence set $Seq(i, x, j)$ and complete-link set $Link(x, j, k)$.

$$Push(x, \mathbf{d}) = (x, d_0, d_1, \ldots, d_{n-2}) \tag{4}$$
$$Cover(x, \mathbf{d}) = (x, d_1, d_2, \ldots, d_{n-1}) \tag{5}$$
$$Insert(x, \mathbf{d}) = (d_0, x, d_2, \ldots, d_{n-1}) \tag{6}$$

Then, the $Link(\mathbf{d})$ and $Seq(\mathbf{d})$ can be defined by Exp. (7) and Exp. (9) below, where "$\times$" indicates the direct product of sets.

$$\begin{aligned} Link(\mathbf{d}) = &\bigcup_{\substack{\text{if } d_1=l+1, \text{ then } i=d_1-1; \\ \text{else } i\in[\min(d_0,d_1),\ \max(d_0,d_1)-1]}} \\ &\{Seq(Left(i, \mathbf{d}))\times \\ &\quad Seq(Right(i + 1, \mathbf{d})) \times \mathbf{d}\} \end{aligned} \tag{7}$$

where

$$(Left, Right) = \begin{cases} (Push, Cover), & \text{if } d_0 < d_1 \\ (Cover, Push), & \text{if } d_0 > d_1 \end{cases} \tag{8}$$

$$\begin{aligned} Seq(\mathbf{d}) = &\bigcup_{\substack{i\in[\min(d_0,d_1),\ \max(d_0,d_1)] \\ \text{and } i\neq d_1}} \\ &\{Seq(Insert(i, \mathbf{d}))\times \\ &\quad Link(Cover(i, \mathbf{d}))\} \end{aligned} \tag{9}$$

Exp. (7) 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.[10] Exp. (9) 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.

To start the recursive definition, we replace d_0 by $\langle /L \rangle$ and $\langle /R \rangle$ for all $Seq(\mathbf{d})$ with $d_0 = d_1$[11].

$$Left(x, \mathbf{d}) = Left(\langle /R \rangle, \mathbf{d}), \quad \text{if } x = \min(d_0, d_1) \text{ in Exp. (7)} \tag{10}$$

$$Right(x, \mathbf{d}) = Right\ (\langle /L \rangle, \mathbf{d}), \quad \text{if } x = \max(d_0, d_1) \text{ in Exp. (7)} \tag{11}$$

$$Insert(x, \mathbf{d}) = Push(\langle /L \rangle, \mathbf{d}), \quad \text{if } x = d_0, \text{ and } d_0 < d_1 \text{ in Exp. (9)} \tag{12}$$

$$Insert(x, \mathbf{d}) = Push(\langle /R \rangle, \mathbf{d}), \quad \text{if } x = d_0, \text{ and } d_0 > d_1 \text{ in Exp. (9)} \tag{13}$$

4.3 Estimation

According to the recursive definition, it is natural to derive an inside-outside algorithm (Lari and Young, 1990), which is an adaption of the EM algorithm (Dempster et al., 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. (7) and Exp. (9) to Exp. (14) and Exp. (15) respectively, where the notation $\langle \cdot, \cdot \rangle$ stands for an unordered 2-tuple of a complete-set pair.

$$Link(\mathbf{d}) = \bigcup_{\forall \langle Sub_1, Sub_2 \rangle} \{Sub_1 \times Sub_2 \times \mathbf{d}\} \tag{14}$$

$$Seq(\mathbf{d}) = \bigcup_{\forall \langle Sub_1, Sub_2 \rangle} \{Sub_1 \times Sub_2\} \tag{15}$$

We further define $R_{\text{Link}}(Link(\mathbf{d}), \langle Sub_1, Sub_2 \rangle)$ as a relation for $Link(\mathbf{d})$, $\langle Sub_1, Sub_2 \rangle$ satisfying Exp. (14). Similarly, $R_{\text{Seq}}(Seq(\mathbf{d}), \langle Sub_1, Sub_2 \rangle)$ is a relation for $Seq(\mathbf{d})$, $\langle Sub_1, Sub_2 \rangle$ satisfying Exp. (15). Then, the inside probability β and outside probability α of the two kinds of complete sets can be calculated by Exp. (16) to Exp. (19), where $p(\mathbf{d})$ is the probability of the lexical dependency N-gram represented by $\mathbf{d}$ in a sentence.

$$\beta(Link(\mathbf{d})) = \sum_{\substack{\langle Sub_1, Sub_2 \rangle, \text{ s.t.} \\ R_{\text{Link}}(Link(\mathbf{d}), \langle Sub_1, Sub_2 \rangle)}} \beta(Sub_1)\beta(Sub_2)p(\mathbf{d}) \tag{16}$$

$$\beta(Seq(\mathbf{d})) = \sum_{\substack{\langle Sub_1, Sub_2 \rangle, \text{ s.t.} \\ R_{\text{Seq}}(Seq(\mathbf{d}), \langle Sub_1, Sub_2 \rangle)}} \beta(Sub_1)\beta(Sub_2) \tag{17}$$

$$\alpha(Link(\mathbf{d})) = \sum_{\substack{\langle Sup, Con \rangle, \text{ s.t.} \\ R_{\text{Seq}}(Sup, \langle Link(\mathbf{d}), Con \rangle)}} \alpha(Sup)\beta(Con) \tag{18}$$

$$\alpha(Seq(\mathbf{d})) = \sum_{\substack{\langle Sup, Con \rangle, \text{ s.t.} \\ R_{\text{Link}}(Sup, \langle Seq(\mathbf{d}), Con \rangle)}} \alpha(Sup)\beta(Con)p(\mathbf{d}') + \sum_{\substack{\langle Sup, Con \rangle, \text{ s.t.} \\ R_{\text{Seq}}(Sup, \langle Seq(\mathbf{d}), Con \rangle)}} \alpha(Sup)\beta(Con) \tag{19}$$

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

Specifically, Exp. (16) and Exp. (17) can be directly derived from the definitions of Exp. (7) and Exp. (9), respectively. Further, a complete-link set can only be a component of a complete-sequence set from Exp. (9), while a complete-sequence set can be both a component of a complete-link set from Exp. (7), and a component of a complete-sequence set from Exp. (9). As a result, Exp. (18) and Exp. (19) can be derived respectively.

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

$$\beta(Seq(\mathbf{d})) = p(\mathbf{d}) \tag{20}$$

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

$$P(S) = \beta(Seq(\mathbf{d} = (1, l+1, 0, \cdots, 0))) \tag{21}$$

For the re-estimation, we can get the probabilistic counts[12] of a dependency N-gram represented by $\mathbf{d}$ in a sentence using:

$$\beta(Link(\mathbf{d})) \cdot \alpha(Link(\mathbf{d})) \cdot P(S)^{-1} \quad [13] \tag{22}$$

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

[10] We further restrict the *root* mark $\langle /s \rangle$ to take only one modifier (the situation when $d_1 = l + 1$ in Exp. (7)), according to the general restrictions of the dependency grammar.

[11] From the restriction in Exp. (2), d_0 should not be equal to d_1. This is only possible for those $Seq(\mathbf{d})$ at the start of the recursive definition, where the word at d_0 is actually a $\langle /L \rangle$ tag or a $\langle /R \rangle$ tag, which does not have an absolute position in a sentence.

[12] They are no longer integers.

[13] For the situation in Exp. (20), we use $\frac{\beta(Seq(\mathbf{d})) \cdot \alpha(Seq(\mathbf{d}))}{P(S)}$.

5 Experiments

5.1 Experiment Setting

Corpus

As the proposed dependency N-gram model and estimation algorithm are language-independent, we conduct experiments using four different languages: English, German, Spanish and Japanese. The corpora we use for English, German, and Spanish are the sets of sentences with $5-15$ words from the corresponding single-language corpora of **Europarl**[14] (Koehn, 2005). The corpus for Japanese is the set of sentences with $5-20$ words from the Japanese side of the **NTCIR-8** corpus (Fujii et al., 2010). We take one two-hundredth 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 and test sets are shown in Tables 1 and 2.

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 1: The training sets.

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

Table 2: The numbers of sentences in development and test sets.

Parameter Collection and Initialization

In order 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 from the raw corpora all possible lexical dependency N-grams[15] without any cut-off thresholds for models of every order. Before estimation, we use relative frequency to initialize the probabilities.

[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 two orientations are considered. The parameter collection and initialization do not take the structure into account.

5.2 Results

Algorithm Convergence

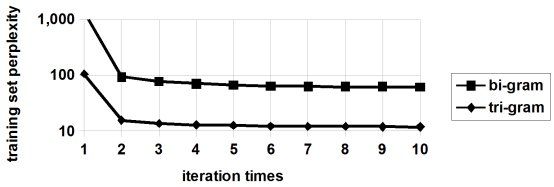

Figure 9: The English training set perplexities before each iteration. (The y-axis is logarithmic.)

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 5 iterations. However, for the dependency tri-gram, the first iteration already reaches a 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 already be too complex a model with too many parameters, so the features of the training set are represented well, resulting in a low perplexity. This suggests the model is over-fitting the data. We discuss this in Sec. 5.3.

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 hand-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. Then 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 also 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 Kneser-Ney discounting to handle the unknown words. The uni-gram model is interpolated with the dependency bi-gram model. Fur-

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

Table 3: The development set perplexities (dev-ppl) and test set perplexities (test-ppl) of dependency N-gram models (N = 2 (bi), 3 (tri)). 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}. $(1-\lambda_{bi})$ 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.

thermore, as the $\langle/L\rangle$ tag and $\langle/R\rangle$ tag are taken as general words but they never really appear in a training set, we treat them separately, and interpolate them with the uni-gram model.

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

Table 4: The test set perplexities of the original N-gram models. MLE is the maximum likelihood estimation realized by setting the *adding delta* to 0 in adding smoothing. MKN is the interpolated modified Kneser-Ney discounting.

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, the other one is constructed by state-of-the-art interpolated modified Kneser-Ney 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 Table 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 Sec. 5.3.

[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 by. Therefore, we only count the lexical words in perplexity calculation for fairness.

5.3 Discussion

Parameter Number

For a sentence with l words, the number of dependency N-gram 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 the sentence length l, a large N will be practically intractable, especially for long sentences. In Fig. 10, we show the numbers of complete sets of different order dependency N-gram models for different sentence lengths.

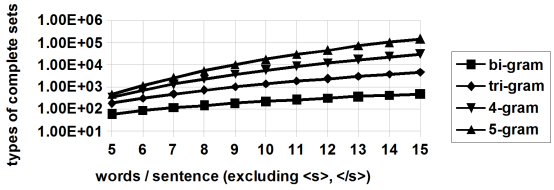

Figure 10: The numbers of complete sets. (The y-axis is logarithmic.)

This behavior is also related to the over-fitting problem because our algorithm is essentially an iterative maximum likelihood estimation. A model that is too complex will be too 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 Fig. 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.

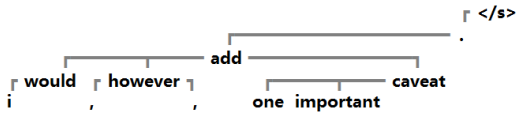

Figure 11: The best dependency structure of the English sentence "*i would , however , add one important caveat .*"

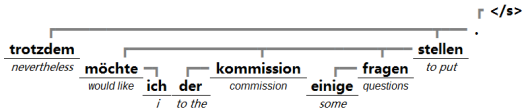

Figure 12: The best dependency structure of the German sentence "*trotzdem möchte ich der kommission einige fragen stellen .*"

Test Set Perplexity

Comparing the test set perplexities in Tables 3 and 4, we can see the dependency bi-gram model achieves the same, or sometimes better performance of the original N-gram language models. However, when we look at the tri-grams, the interpolated modified Kneser-Ney (KN) discounting method, which is 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 uses an efficient discounting to avoid the over-fitting problem, and our model has no smoothing, the difference in performance is reasonable for complex models. On the other hand, the generally 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.

Model Preference

In Fig. 11 to Fig. 14, we present examples of the best dependency structures generated by our approach of sentences in test sets. We used the settings in Table 3 and generated them by the Viterbi algorithm (Viterbi, 1967). It can be seen the proposed approach can reveal features of specific languages even though it is unsupervised: e.g. the final-position verb "*stellen*" and its relation with the second-position auxiliary verb "*möchte*" in the German sentence. 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. For example, in the Spanish sentence, syntactically the preposition "*a*"

[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.

[19]Articles, prepositions, etc.

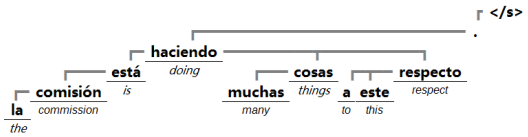

Figure 13: The best dependency structure of the Spanish sentence "*la comisión está haciendo muchas cosas a este respecto .*"

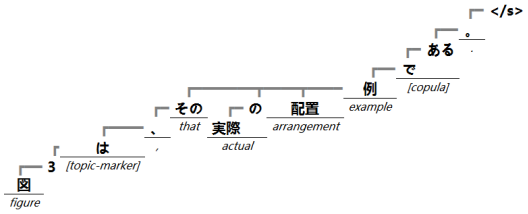

Figure 14: The best dependency structure of the Japanese sentence "図 *3* は 、 その 実際 の 配置 例 で ある 。"

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: "*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[20].

6 Conclusion and Future Work

In this paper, we proposed a generative dependency N-gram language model and the definition of the complete sets for arbitrary order, by which an unsupervised parameter estimation algorithm is facilitated. The experimental results demonstrate the applicability and the properties of the proposed approach. In future work, we will develop methods of parameter pruning and discounting to handle the over-fitting problem. As the proposed dependency language model is intrinsically complex, we also plan to do more fundamental simplifications. On the other hand, although our proposed algorithm is unsupervised, the output of a trained parser, which can provide clear and lexical heuristics, can be integrated in it. We will investigate this possibility and evaluate the performance by linguistically motivated criteria.

Acknowledgment

We would like to thank anonymous reviewers for their valuable comments and suggestions. This work was supported by JSPS KAKENHI Grant Number 24650063.

[20]This tendency, however, is correct for articles, such as the "*der*" in German and "*la*" in Spanish.

References

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vicent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283–332.

Stanley F. Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical report, TR-10-98, Computer Science Group, Harvard Univ.

Michael Collins. 1998. Three generative, lexicalised models for statistical parsing. In *Proc. of ACL 1998*, pages 16–23.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin . 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: an exploration. In *Proc. of COLING 1996*, pages 340–345.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, Takehito Utsuro, Terumasa Ehara, Hiroshi Echizenya, and Sayori Shimohata. 2010. Overview of the patent translation task at the NTCIR-8 workshop. In *Proc. of NTCIR-8*, pages 371–376.

Jianfeng Gao and Hisami Suzuki. 2003. Unsupervised learning of dependency structure for language modeling. In *Proc. of ACL 2003*, pages 521–580.

Yvette Graham and Josef van Genabith. 2010. Deep syntax language models and statistical machine translation. In *Proc. of SSST at COLING 2010*, pages 118–126.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL 2003*, pages 423–430.

Dan Klein and Chirstopher D. Manning. 2004. Corpus-based induction of syntactic structure: models of dependency and constituency. In *Proc. of ACL 2004*, pages 478–485.

Philipp Koehn. 2005. Europarl: a parallel corpus for statistical machine translation. In *Proc. of MT summit 2005*, pages 79–86.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proc. of COLING 2002*, pages 1–7.

K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4(1):35–56.

Seungmi Lee and Key-Sun Choi. 1997. Reestimation and best-first parsing algorithm for probabilistic dependency grammars. In *Proc. of WVLC 1997*, pages 41–55.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Mark A. Paskin. 2002. Grammatical digrams. *Advances In Neural Information Processing Systems 14*, 1:91–97.

Andreas Stolcke, Ciprian Chelba, David Engle, Victor Jimenez, Lidia Mangu, Harry Printz, Eric Ristad, Roni Rosenfeld, Dekai Wu, Frederick Jelinek, and Sanjeev Khudanpur. 1997. Dependency language modeling.

Andreas Stolcke. 2002. SRILM–an extensible language modeling toolkit. In *Proc. of ICSLP 2002*, pages 901–904.

Christoph Tillmann and Hermann Ney. 1997. Word triggers and the EM algorithm. In *Proc. of CoNNL 1997*, pages 117–124.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on Information Theory*, 13(2):260–269.