Unsupervised Learning of Dependency Structure for Language Modeling

Jianfeng Gao
Microsoft Research, Asia
49 Zhichun Road, Haidian District
Beijing 100080 China
jfgao@microsoft.com

Hisami Suzuki
Microsoft Research
One Microsoft Way
Redmond WA 98052 USA
hisamis@microsoft.com

Abstract

This paper presents a dependency language model (DLM) that captures linguistic constraints via a dependency structure, i.e., a set of probabilistic dependencies that express the relations between headwords of each phrase in a sentence by an acyclic, planar, undirected graph. Our contributions are three-fold. First, we incorporate the dependency structure into an *n*-gram language model to capture long distance word dependency. Second, we present an unsupervised learning method that discovers the dependency structure of a sentence using a bootstrapping procedure. Finally, we evaluate the proposed models on a realistic application (Japanese Kana-Kanji conversion). Experiments show that the best DLM achieves an 11.3% error rate reduction over the word trigram model.

1 Introduction

In recent years, many efforts have been made to utilize linguistic structure in language modeling, which for practical reasons is still dominated by trigram-based language models. There are two major obstacles to successfully incorporating linguistic structure into a language model: (1) capturing longer distance word dependencies leads to higher-order *n*-gram models, where the number of parameters is usually too large to estimate; (2) capturing deeper linguistic relations in a language model requires a large annotated training corpus and a decoder that assigns linguistic structure, which are not always available.

This paper presents a new dependency language model (DLM) that captures long distance linguistic constraints between words via a dependency structure, i.e., a set of probabilistic dependencies that capture linguistic relations between headwords of each phrase in a sentence. To deal with the first obstacle mentioned above, we approximate long-distance linguistic dependency by a model that is similar to a skipping bigram model in which the prediction of a word is conditioned on exactly one other linguistically related word that lies arbitrarily far in the past. This dependency model is then interpolated with a headword bigram model and a word trigram model, keeping the number of parameters of the combined model manageable. To overcome the second obstacle, we used an unsupervised learning method that discovers the dependency structure of a given sentence using an Expectation-Maximization (EM)-like procedure. In this method, no manual syntactic annotation is required, thereby opening up the possibility for building a language model that performs well on a wide variety of data and languages. The proposed model is evaluated using Japanese Kana-Kanji conversion, achieving significant error rate reduction over the word trigram model.

2 Motivation

A trigram language model predicts the next word based only on two preceding words, blindly discarding any other relevant word that may lie three or more positions to the left. Such a model is likely to be linguistically implausible: consider the English sentence in Figure 1(a), where a trigram model would predict *cried* from *next seat*, which does not agree with our intuition. In this paper, we define a dependency structure of a sentence as a set of probabilistic dependencies that express linguistic relations between words in a sentence by an acyclic, planar graph, where two related words are connected by an undirected graph edge (i.e., we do not differentiate the modifier and the head in a dependency). The dependency structure for the sentence in Figure 1(a) is as shown; a model that uses this dependency structure would predict *cried* from *baby*, in agreement with our intuition.

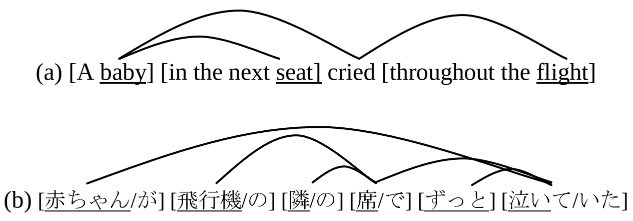
(a) [A baby] [in the next seat] cried [throughout the flight]
(b) [赤ちゃん/が] [飛行機/の] [隣/の] [席/で] [ずっと] [泣いて/いた]

Figure 1. Examples of dependency structure. **(a)** A dependency structure of an English sentence. Square brackets indicate base NPs; underlined words are the headwords. **(b)** A Japanese equivalent of (a). Slashes demarcate morpheme boundaries; square brackets indicate phrases (*bunsetsu*).

A Japanese sentence is typically divided into non-overlapping phrases called *bunsetsu*. As shown in Figure 1(b), each bunsetsu consists of one content word, referred to here as the *headword H*, and several function words *F*. Words (more precisely, morphemes) within a bunsetsu are tightly bound with each other, which can be adequately captured by a word trigram model. However, headwords across bunsetsu boundaries also have dependency relations with each other, as the diagrams in Figure 1 show. Such long distance dependency relations are expected to provide useful and complementary information with the word trigram model in the task of next word prediction.

In constructing language models for realistic applications such as speech recognition and Asian language input, we are faced with two constraints that we would like to satisfy: First, the model must operate in a left-to-right manner, because (1) the search procedures for predicting words that correspond to the input acoustic signal or phonetic string work left to right, and (2) it can be easily combined with a word trigram model in decoding. Second, the model should be computationally feasible both in training and decoding. In the next section, we offer a DLM that satisfies both of these constraints.

3 Dependency Language Model

The DLM attempts to generate the dependency structure incrementally while traversing the sentence left to right. It will assign a probability to every word sequence *W* and its dependency structure *D*. The probability assignment is based on an encoding of the (*W*, *D*) pair described below.

Let *W* be a sentence of length *n* words to which we have prepended `<s>` and appended `</s>` so that w_0 = `<s>`, and w_{n+1} = `</s>`. In principle, a language model recovers the probability of a sentence $P(W)$ over all possible *D* given *W* by estimating the joint probability $P(W, D)$: $P(W) = \sum_D P(W, D)$. In practice, we used the so-called maximum approximation where the sum is approximated by a single term $P(W, D^*)$:

$$P(W) = \sum_D P(W,D) \approx P(W,D^*)\,. \tag{1}$$

Here, D^* is the most probable dependency structure of the sentence, which is generally discovered by maximizing $P(W, D)$:

$$D^* = \arg\max_D P(W,D)\,. \tag{2}$$

Below we restrict the discussion to the most probable dependency structure of a given sentence, and simply use *D* to represent D^*. In the remainder of this section, we first present a statistical dependency parser, which estimates the parsing probability at the word level, and generates *D* incrementally while traversing *W* left to right. Next, we describe the elements of the DLM that assign probability to each possible *W* and its most probable *D*, $P(W, D)$. Finally, we present an EM-like iterative method for unsupervised learning of dependency structure.

3.1 Dependency parsing

The aim of dependency parsing is to find the most probable *D* of a given *W* by maximizing the probability $P(D|W)$. Let *D* be a set of probabilistic dependencies *d*, i.e. $d \in D$. Assuming that the dependencies are independent of each other, we have

$$P(D|W) = \prod_{d \in D} P(d|W) \tag{3}$$

where $P(d|W)$ is the dependency probability conditioned by a particular sentence.[1] It is impossible to estimate $P(d|W)$ directly because the same sentence is very unlikely to appear in both training and test data. We thus approximated $P(d|W)$ by $P(d)$, and estimated the dependency probability from the training corpus. Let $d_{ij} = (w_i, w_j)$ be the dependency

[1] The model in Equation (3) is not strictly probabilistic because it drops the probabilities of illegal dependencies (e.g., crossing dependencies).

between w_i and w_j. The maximum likelihood estimation (MLE) of $P(d_{ij})$ is given by

$$P(d_{ij}) = \frac{C(w_i, w_j, R)}{C(w_i, w_j)} \qquad (4)$$

where $C(w_i, w_j, R)$ is the number of times w_i and w_j have a dependency relation in a sentence in training data, and $C(w_i, w_j)$ is the number of times w_i and w_j are seen in the same sentence. To deal with the data sparseness problem of MLE, we used the backoff estimation strategy similar to the one proposed in Collins (1996), which backs off to estimates that use less conditioning context. More specifically, we used the following three estimates:

$$E_1 = \frac{\eta_1}{\delta_1} \quad E_{23} = \frac{\eta_2 + \eta_3}{\delta_2 + \delta_3} \qquad E_4 = \frac{\eta_4}{\delta_4}, \qquad (5)$$

Where

$$\eta_1 = C(w_i, w_j, R),\ \delta_1 = C(w_i, w_j),$$
$$\eta_2 = C(w_i, *, R),\ \delta_2 = C(w_i, *),$$
$$\eta_3 = C(*, w_j, R),\ \delta_3 = C(*, w_j),$$
$$\eta_4 = C(*, *, R),\ \delta_4 = C(*, *).$$

in which * indicates a wild-card matching any word. The final estimate E is given by linearly interpolating these estimates:

$$E = \lambda_1 E_1 + (1-\lambda_1)(\lambda_2 E_{23} + (1-\lambda_2)E_4) \qquad (6)$$

where λ_1 and λ_2 are smoothing parameters.

Given the above parsing model, we used an approximation parsing algorithm that is $O(n^2)$. Traditional techniques use an optimal Viterbi-style algorithm (e.g., bottom-up chart parser) that is $O(n^5)$.[2] Although the approximation algorithm is not guaranteed to find the most probable D, we opted for it because it works in a left-to-right manner, and is very efficient and simple to implement. In our experiments, we found that the algorithm performs reasonably well on average, and its speed and simplicity make it a better choice in DLM training where we need to parse a large amount of training data iteratively, as described in Section 3.3.

The parsing algorithm is a slightly modified version of that proposed in Yuret (1998). It reads a sentence left to right; after reading each new word w_j, it tries to link w_j to each of its previous words w_i, and push the generated dependency d_{ij} into a stack. When a dependency crossing or a cycle is detected in the stack, the dependency with the lowest dependency probability in conflict is eliminated. The algorithm is outlined in Figures 2 and 3.

```
DEPENDENCY-PARSING($W$)
1  for $j \leftarrow 1$ to LENGTH($W$)
2      for $i \leftarrow j-1$ downto $1$
3          PUSH $d_{ij} = (w_i, w_j)$ into the stack $D_j$
4          if a dependency cycle ($CY$) is detected in $D_j$
           (see Figure 3(a))
5              REMOVE $d$, where $d = \arg\min_{d \in CY} P(d)$
6          while a dependency crossing ($CR$) is detected
           in $D_j$ (see Figure 3(b)) do
7              REMOVE $d$, where $d = \arg\min_{d \in CR} P(d)$
8  OUTPUT($D$)
```

Figure 2. Approximation algorithm of dependency parsing

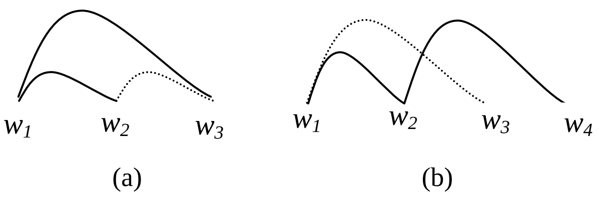

Figure 3. (a) An example of a dependency cycle: given that $P(d_{23})$ is smaller than $P(d_{12})$ and $P(d_{13})$, d_{23} is removed (represented as dotted line). **(b)** An example of a dependency crossing: given that $P(d_{13})$ is smaller than $P(d_{24})$, d_{13} is removed.

Let the dependency probability be the measure of the strength of a dependency, i.e., higher probabilities mean stronger dependencies. Note that when a strong new dependency crosses multiple weak dependencies, the weak dependencies are removed even if the new dependency is weaker than the sum of the old dependencies.[3] Although this action results in lower total probability, it was implemented because multiple weak dependencies connected to the beginning of the sentence often pre-

[2] For parsers that use bigram lexical dependencies, Eisner and Satta (1999) presents parsing algorithms that are $O(n^4)$ or $O(n^3)$. We thank Joshua Goodman for pointing this out.

[3] This operation leaves some headwords disconnected; in such a case, we assumed that each disconnected headword has a dependency relation with its preceding headword.

vented a strong meaningful dependency from being created. In this manner, the directional bias of the approximation algorithm was partially compensated for.[4]

3.2 Language modeling

The DLM together with the dependency parser provides an encoding of the (*W*, *D*) pair into a sequence of elementary model actions. Each action conceptually consists of two stages. The first stage assigns a probability to the next word given the left context. The second stage updates the dependency structure given the new word using the parsing algorithm in Figure 2. The probability *P*(*W*, *D*) is calculated as:

$$P(W,D) = \prod_{j=1}^{n} P(w_j \mid \Phi(W_{j-1}, D_{j-1})) P(D_{j-1}^{j} \mid \Phi(W_{j-1}, D_{j-1}), w_j) \tag{7}$$

$$P(D_{j-1}^{j} \mid \Phi(W_{j-1}, D_{j-1}), w_j) = \prod_{i=1}^{j} P(p_i^j \mid W_{j-1}, D_{j-1}, p_1^j, \ldots, p_{i-1}^j)\,. \tag{8}$$

Here (W_{j-1}, D_{j-1}) is the word-parse (*j*-1)-prefix that D_{j-1} is a dependency structure containing only those dependencies whose two related words are included in the word (*j*-1)-prefix, W_{j-1}. w_j is the word to be predicted. D_{j-1}^{j} is the incremental dependency structure that generates $D_j = D_{j-1} \,||\, D_{j-1}^{j}$ (|| stands for concatenation) when attached to D_{j-1}; it is the dependency structure built on top of D_{j-1} and the newly predicted word w_j (see the **for**-loop of line 2 in Figure 2). p_i^j denotes the *i*th action of the parser at position *j* in the word string: to generate a new dependency d_{ij}, and eliminate dependencies with the lowest dependency probability in conflict (see lines 4 – 7 in Figure 2). Φ is a function that maps the history (W_{j-1}, D_{j-1}) onto equivalence classes.

The model in Equation (8) is unfortunately infeasible because it is extremely difficult to estimate the probability of p_i^j due to the large number of parameters in the conditional part. According to the parsing algorithm in Figure 2, the probability of each action p_i^j depends on the entire history (e.g. for detecting a dependency crossing or cycle), so any mapping Φ that limits the equivalence classification to less context suitable for model estimation would be very likely to drop critical conditional information for predicting p_i^j. In practice, we approximated $P(D_{j-1}^{j} \mid \Phi(W_{j-1}, D_{j-1}), w_j)$ by $P(D_j|W_j)$ of Equation (3), yielding $P(W_j, D_j) \approx P(W_j \mid \Phi(W_{j-1}, D_{j-1}))\, P(D_j|W_j)$. This approximation is probabilistically deficient, but our goal is to apply the DLM to a decoder in a realistic application, and the performance gain achieved by this approximation justifies the modeling decision.

Now, we describe the way $P(w_j|\Phi(W_{j-1},D_{j-1}))$ is estimated. As described in Section 2, headwords and function words play different syntactic and semantic roles capturing different types of dependency relations, so the prediction of them can better be done separately. Assuming that each word token can be uniquely classified as a headword or a function word in Japanese, the DLM can be conceived of as a cluster-based language model with two clusters, headword *H* and function word *F*. We can then define the conditional probability of w_j based on its history as the product of two factors: the probability of the category given its history, and the probability of w_j given its category. Let h_j or f_j be the actual headword or function word in a sentence, and let H_j or F_j be the category of the word w_j. $P(w_j|\Phi(W_{j-1},D_{j-1}))$ can then be formulated as:

$$\begin{aligned} P(w_j \mid \Phi(W_{j-1}, D_{j-1})) = {} & P(H_j \mid \Phi(W_{j-1}, D_{j-1})) \times P(w_j \mid \Phi(W_{j-1}, D_{j-1}), H_j) \\ & + P(F_j \mid \Phi(W_{j-1}, D_{j-1})) \times P(w_j \mid \Phi(W_{j-1}, D_{j-1}), F_j)\,. \end{aligned} \tag{9}$$

We first describe the estimation of headword probability $P(w_j \mid \Phi(W_{j-1}, D_{j-1}), H_j)$. Let HW_{j-1} be the headwords in (*j*-1)-prefix, i.e., containing only those headwords that are included in W_{j-1}. Because HW_{j-1} is determined by W_{j-1}, the headword probability can be rewritten as $P(w_j \mid \Phi(W_{j-1}, HW_{j-1}, D_{j-1}), H_j)$. The problem is to determine the mapping Φ so as to identify the related words in the left context that we would like to condition on. Based on the discussion in Section 2, we chose a mapping function that retains (1) two preceding words w_{j-1} and w_{j-2} in W_{j-1}, (2) one preceding headword h_{j-1} in HW_{j-1}, and (3) one linguistically related word w_i according to D_{j-1}. w_i is determined in two stages: First, the parser updates the dependency structure

[4] Theoretically, we should arrive at the same dependency structure no matter whether we parse the sentence left to right or right to left. However, this is not the case with the approximation algorithm. This problem is called *directional bias*.

D_{j-1} incrementally to D_j assuming that the next word is w_j. Second, when there are multiple words that have dependency relations with w_j in D_j, w_i is selected using the following decision rule:

$$w_i = \underset{w_i:(w_i,w_j)\in D_j}{\arg\max}\ P(w_j \mid w_i, R), \tag{10}$$

where the probability $P(w_j \mid w_i, R)$ of the word w_j given its linguistic related word w_i is computed using MLE by Equation (11):

$$P(w_j \mid w_i, R) = \frac{C(w_i, w_j, R)}{\sum_{w_j} C(w_i, w_j, R)}. \tag{11}$$

We thus have the mapping function $\Phi(W_{j-1}, HW_{j-1}, D_{j-1}) = (w_{j-2}, w_{j-1}, h_{j-1}, w_i)$. The estimate of headword probability is an interpolation of three probabilities:

$$\begin{aligned} P(w_j \mid \Phi(W_{j-1}, D_{j-1}), H_j) = & \\ & \lambda_1(\lambda_2 P(w_j \mid h_{j-1}, H_j) \\ & + (1-\lambda_2)P(w_j \mid w_i, R)) \\ & + (1-\lambda_1)P(w_j \mid w_{j-2}, w_{j-1}, H_j). \end{aligned} \tag{12}$$

Here $P(w_j|w_{j-2}, w_{j-1}, H_j)$ is the word trigram probability given that w_j is a headword, $P(w_j|h_{j-1}, H_j)$ is the headword bigram probability, and $\lambda_1, \lambda_2 \in [0,1]$ are the interpolation weights optimized on held-out data.

We now come back to the estimate of the other three probabilities in Equation (9). Following the work in Gao et al. (2002b), we used the unigram estimate for word category probabilities, (i.e., $P(H_j|\Phi(W_{j-1}, D_{j-1})) \approx P(H_j)$ and $P(F_j \mid \Phi(W_{j-1}, D_{j-1})) \approx P(F_j)$), and the standard trigram estimate for function word probability (i.e., $P(w_j \mid \Phi(W_{j-1},D_{j-1}),F_j) \approx P(w_j \mid w_{j-2}, w_{j-1}, F_j)$). Let C_j be the category of w_j; we approximated $P(C_j)\times P(w_j|w_{j-2}, w_{j-1}, C_j)$ by $P(w_j \mid w_{j-2}, w_{j-1})$. By separating the estimates for the probabilities of headwords and function words, the final estimate is given below:

$$P(w_j \mid \Phi(W_{j-1}, D_{j-1})) = \tag{13}$$

$$\begin{cases} \lambda_1(P(H_j)(\lambda_2 P(w_j \mid h_{j-1}) \\ \quad + (1-\lambda_2)P(w_j \mid w_i, R)) \\ \quad + (1-\lambda_1)P(w_j \mid w_{j-2}, w_{j-1}) \\ \qquad w_j\text{: headword} \\ P(w_j \mid w_{j-2}, w_{j-1}) \\ \qquad w_j\text{: function word} \end{cases}$$

All conditional probabilities in Equation (13) are obtained using MLE on training data. In order to deal with the data sparseness problem, we used a backoff scheme (Katz, 1987) for parameter estimation. This backoff scheme recursively estimates the probability of an unseen n-gram by utilizing (n–1)-gram estimates. In particular, the probability of Equation (11) backs off to the estimate of $P(w_j|R)$, which is computed as:

$$P(w_j \mid R) = \frac{C(w_j, R)}{N}, \tag{14}$$

where N is the total number of dependencies in training data, and $C(w_j, R)$ is the number of dependencies that contains w_j. To keep the model size manageable, we removed all n-grams of count less than 2 from the headword bigram model and the word trigram model, but kept all long-distance dependency bigrams that occurred in the training data.

3.3 Training data creation

This section describes two methods that were used to tag raw text corpus for DLM training: (1) a method for headword detection, and (2) an unsupervised learning method for dependency structure acquisition.

In order to classify a word uniquely as H or F, we used a mapping table created in the following way. We first assumed that the mapping from part-of-speech (POS) to word category is unique and fixed;[5] we then used a POS-tagger to generate a POS-tagged corpus, which are then turned into a category-tagged corpus.[6] Based on this corpus, we created a mapping table which maps each word to a unique category: when a word can be mapped to either H or F, we chose the more frequent category in the corpus. This method achieved a 98.5% accuracy of headword detection on the test data we used.

Given a headword-tagged corpus, we then used an EM-like iterative method for joint optimization of the parsing model and the dependency structure of training data. This method uses the maximum likelihood principle, which is consistent with lan-

[5] The tag set we used included 1,187 POS tags, of which 102 counted as headwords in our experiments.

[6] Since the POS-tagger does not identify phrases (*bunsetsu*), our implementation identifies multiple headwords in phrases headed by compounds.

guage model training. There are three steps in the algorithm: (1) initialize, (2) (re-)parse the training corpus, and (3) re-estimate the parameters of the parsing model. Steps (2) and (3) are iterated until the improvement in the probability of training data is less than a threshold.

Initialize: We set a window of size *N* and assumed that each headword pair within a headword *N*-gram constitutes an initial dependency. The optimal value of *N* is 3 in our experiments. That is, given a headword trigram (h_1, h_2, h_3), there are 3 initial dependencies: d_{12}, d_{13}, and d_{23}. From the initial dependencies, we computed an initial dependency parsing model by Equation (4).

(Re-)parse the corpus: Given the parsing model, we used the parsing algorithm in Figure 2 to select the most probable dependency structure for each sentence in the training data. This provides an updated set of dependencies.

Re-estimate the parameters of parsing model: We then re-estimated the parsing model parameters based on the updated dependency set.

4 Evaluation Methodology

In this study, we evaluated language models on the application of Japanese Kana-Kanji conversion, which is the standard method of inputting Japanese text by converting the text of a syllabary-based Kana string into the appropriate combination of Kanji and Kana. This is a similar problem to speech recognition, except that it does not include acoustic ambiguity. Performance on this task is measured in terms of the character error rate (CER), given by the number of characters wrongly converted from the phonetic string divided by the number of characters in the correct transcript.

For our experiments, we used two newspaper corpora, Nikkei and Yomiuri Newspapers, both of which have been pre-word-segmented. We built language models from a 36-million-word subset of the Nikkei Newspaper corpus, performed parameter optimization on a 100,000-word subset of the Yomiuri Newspaper (held-out data), and tested our models on another 100,000-word subset of the Yomiuri Newspaper corpus. The lexicon we used contains 167,107 entries.

Our evaluation was done within a framework of so-called “N-best rescoring” method, in which a list of hypotheses is generated by the baseline language model (a word trigram model in this study), which is then rescored using a more sophisticated language model. We use the N-best list of N=100, whose “oracle” CER (i.e., the CER of the hypotheses with the minimum number of errors) is presented in Table 1, indicating the upper bound on performance. We also note in Table 1 that the performance of the conversion using the baseline trigram model is much better than the state-of-the-art performance currently available in the marketplace, presumably due to the large amount of training data we used, and to the similarity between the training and the test data.

Baseline Trigram	Oracle of 100-best
3.73%	1.51%

Table 1. CER results of baseline and 100-best list

5 Results

The results of applying our models to the task of Japanese Kana-Kanji conversion are shown in Table 2. The baseline result was obtained by using a conventional word trigram model (WTM).[7] HBM stands for headword bigram model, which does not use any dependency structure (i.e. $\lambda_2 = 1$ in Equation (13)). DLM_1 is the DLM that does not use headword bigram (i.e. $\lambda_2 = 0$ in Equation (13)). DLM_2 is the model where the headword probability is estimated by interpolating the word trigram probability, the headword bigram probability, and the probability given one previous linguistically related word in the dependency structure.

Although Equation (7) suggests that the word probability $P(w_j|\Phi(W_{j-1},D_{j-1}))$ and the parsing model probability can be combined through simple multiplication, some weighting is desirable in practice, especially when our parsing model is estimated using an approximation by the parsing score $P(D|W)$. We therefore introduced a parsing model weight *PW*: both DLM_1 and DLM_2 models were built with and without *PW*. In Table 2, the PW- prefix refers to the DLMs with *PW* = 0.5, and the DLMs without PW- prefix refers to DLMs with *PW* = 0. For both DLM_1 and DLM_2, models with the parsing weight achieve better performance; we

[7] For a detailed description of the baseline trigram model, see Gao et al. (2002a).

therefore discuss only DLMs with the parsing weight for the rest of this section.

Model	λ_1	λ_2	CER	CER reduction
WTM	----	----	3.73%	----
HBM	0.2	1	3.40%	8.8%
DLM_1	0.1	0	3.48%	6.7%
PW-DLM_1	0.1	0	3.44%	7.8%
DLM_2	0.3	0.7	3.33%	10.7%
PW-DLM_2	0.3	0.7	3.31%	11.3%

Table 2. Comparison of CER results

By comparing both HBM and PW-LDM_1 models with the baseline model, we can see that the use of headword dependency contributes greatly to the CER reduction: HBM outperformed the baseline model by 8.8% in CER reduction, and PW-LDM_1 by 7.8%. By combining headword bigram and dependency structure, we obtained the best model PW-DLM_2 that achieves 11.3% CER reduction over the baseline. The improvement achieved by PW-DLM_2 over the HBM is statistically significant according to the *t* test (P<0.01). These results demonstrate the effectiveness of our parsing technique and the use of dependency structure for language modeling.

6 Discussion

In this section, we relate our model to previous research and discuss several factors that we believe to have the most significant impact on the performance of DLM. The discussion includes: (1) the use of DLM as a parser, (2) the definition of the mapping function Φ, and (3) the method of unsupervised dependency structure acquisition.

One basic approach to using linguistic structure for language modeling is to extend the conventional language model $P(W)$ to $P(W, T)$, where T is a parse tree of W. The extended model can then be used as a parser to select the most likely parse by $T^* = \text{argmax}_T\ P(W, T)$. Many recent studies (e.g., Chelba and Jelinek, 2000; Charniak, 2001; Roark, 2001) adopt this approach. Similarly, dependency-based models (e.g., Collins, 1996; Chelba et al., 1997) use a dependency structure D of W instead of a parse tree T, where D is extracted from syntactic trees. Both of these models can be called *grammar-based models*, in that they capture the syntactic structure of a sentence, and the model parameters are estimated from syntactically annotated corpora such as the Penn Treebank. DLM, on the other hand, is a *non-grammar-based model*, because it is not based on any syntactic annotation: the dependency structure used in language modeling was learned directly from data in an unsupervised manner, subject to two weak syntactic constraints (i.e., dependency structure is acyclic and planar).[8] This resulted in capturing the dependency relations that are not precisely syntactic in nature within our model. For example, in the conversion of the string below, the word 晩 *ban* 'evening' was correctly predicted in DLM by using the long-distance bigram 朝~晩 *asa~ban* 'morning~evening', even though these two words are not in any direct syntactic dependency relationship:

主席の肖像の前で朝、指示を仰ぎ、晩に報告を行う
'asks for instructions in the morning and submits daily reports in the evening'

Though there is no doubt that syntactic dependency relations provide useful information for language modeling, the most linguistically related word in the previous context may come in various linguistic relations with the word being predicted, not limited to syntactic dependency. This opens up new possibilities for exploring the combination of different knowledge sources in language modeling.

Regarding the function Φ that maps the left context onto equivalence classes, we used a simple approximation that takes into account only one linguistically related word in left context. An alternative is to use the maximum entropy (ME) approach (Rosenfeld, 1994; Chelba et al., 1997). Although ME models provide a nice framework for incorporating arbitrary knowledge sources that can be encoded as a large set of constraints, training and using ME models is extremely computationally expensive. Our working hypothesis is that the information for predicting the new word is dominated by a very limited set of words which can be selected heuristically: in this paper, Φ is defined as a heuristic function that maps D to one word in D that has the strongest linguistic relation with the word being predicted, as in (8). This hypothesis is borne out by

[8] In this sense, our model is an extension of a dependency-based model proposed in Yuret (1998). However, this work has not been evaluated as a language model with error rate reduction.

an additional experiment we conducted, where we used *two* words from *D* that had the strongest relation with the word being predicted; this resulted in a very limited gain in CER reduction of 0.62%, which is not statistically significant (P>0.05 according to the *t* test).

The EM-like method for learning dependency relations described in Section 3.3 has also been applied to other tasks such as hidden Markov model training (Rabiner, 1989), syntactic relation learning (Yuret, 1998), and Chinese word segmentation (Gao et al., 2002a). In applying this method, two factors need to be considered: (1) how to initialize the model (i.e. the value of the window size *N*), and (2) the number of iterations. We investigated the impact of these two factors empirically on the CER of Japanese Kana-Kanji conversion. We built a series of DLMs using different window size *N* and different number of iterations. Some sample results are shown in Table 3: the improvement in CER begins to saturate at the second iteration. We also find that a larger *N* results in a better initial model but makes the following iterations less effective. The possible reason is that a larger *N* generates more initial dependencies and would lead to a better initial model, but it also introduces noise that prevents the initial model from being improved. All DLMs in Table 2 are initialized with N = 3 and are run for two iterations.

Iteration	N = 2	N = 3	N = 5	N = 7	N = 10
Init.	3.552%	3.523%	3.540%	3.514 %	3.511%
1	3.531%	3.503%	3.493%	3.509%	3.489%
2	3.527%	3.481%	3.483%	3.492%	3.488%
3	3.526%	3.481%	3.485%	3.490%	3.488%

Table 3. CER of DLM_1 models initialized with different window size *N*, for 0-3 iterations

7 Conclusion

We have presented a dependency language model that captures linguistic constraints via a dependency structure – a set of probabilistic dependencies that express the relations between headwords of each phrase in a sentence by an acyclic, planar, undirected graph. Promising results of our experiments suggest that long-distance dependency relations can indeed be successfully exploited for the purpose of language modeling.

There are many possibilities for future improvements. In particular, as discussed in Section 6, syntactic dependency structure is believed to capture useful information for informed language modeling, yet further improvements may be possible by incorporating non-syntax-based dependencies. Correlating the accuracy of the dependency parser as a parser vs. its utility in CER reduction may suggest a useful direction for further research.

Reference

Charniak, Eugine. 2001. Immediate-head parsing for language models. In *ACL/EACL 2001*, pp.124-131.

Chelba, Ciprian and Frederick Jelinek. 2000. Structured Language Modeling. *Computer Speech and Language*, Vol. 14, No. 4. pp 283-332.

Chelba, C, D. Engle, F. Jelinek, V. Jimenez, S. Khudanpur, L. Mangu, H. Printz, E. S. Ristad, R. Rosenfeld, A. Stolcke and D. Wu. 1997. Structure and performance of a dependency language model. In *Processing of Eurospeech*, Vol. 5, pp 2775-2778.

Collins, Michael John. 1996. A new statistical parser based on bigram lexical dependencies. In *ACL 34:*184-191.

Eisner, Jason and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *ACL 37*: 457-464.

Gao, Jianfeng, Joshua Goodman, Mingjing Li and Kai-Fu Lee. 2002a. Toward a unified approach to statistical language modeling for Chinese. *ACM Transactions on Asian Language Information Processing*, 1-1: 3-33.

Gao, Jianfeng, Hisami Suzuki and Yang Wen. 2002b. Exploiting headword dependency and predictive clustering for language modeling. In *EMNLP 2002*: 248-256.

Katz, S. M. 1987. Estimation of probabilities from sparse data for other language component of a speech recognizer. *IEEE transactions on Acoustics, Speech and Signal Processing*, 35(3): 400-401.

Rabiner, Lawrence R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of IEEE* 77:257-286.

Roark, Brian. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 17-2: 1-28.

Rosenfeld, Ronald. 1994. Adaptive statistical language modeling: a maximum entropy approach. Ph.D. thesis, Carnegie Mellon University.

Yuret, Deniz. 1998. Discovery of linguistic relations using lexical attraction. Ph.D. thesis, MIT.