Automatic Acquisition of Language Model based on Head-Dependent Relation between Words

Seungmi Lee and **Key-Sun Choi**
Department of Computer Science
Center for Artificial Intelligence Research
Korea Advanced Institute of Science and Technology
e-mail: {leesm, kschoi}@world.kaist.ac.kr

Abstract

Language modeling is to associate a sequence of words with *a priori* probability, which is a key part of many natural language applications such as speech recognition and statistical machine translation. In this paper, we present a language modeling based on a kind of simple dependency grammar. The grammar consists of head-dependent relations between words and can be learned automatically from a raw corpus using the reestimation algorithm which is also introduced in this paper. Our experiments show that the proposed model performs better than n-gram models at 11% to 11.5% reductions in test corpus entropy.

1 Introduction

Language modeling is to associate *a priori* probability to a sentence. It is a key part of many natural language applications such as speech recognition and statistical machine translation. Previous works for language modeling can be broadly divided into two approaches; one is n-gram-based and the other is grammar-based.

N-gram model estimates the probability of a sentence as the product of the probability of each word in the sentence. It assumes that probability of the nth word is dependent on the previous $n-1$ words. The n-gram probabilities are estimated by simply counting the n-gram frequencies in a training corpus. In some cases, class (or part of speech) n-grams are used instead of word n-grams(Brown et al., 1992; Chang and Chen, 1996). N-gram model has been widely used so far, but it has always been clear that n-gram can not represent long distance dependencies.

In contrast with n-gram model, grammar-based approach assigns syntactic structures to a sentence and computes the probability of the sentence using the probabilities of the structures. Long distance dependencies can be represented well by means of the structures. The approach usually makes use of phrase structure grammars such as probabilistic context-free grammar and recursive transition network(Lari and Young, 1991; Sneff, 1992; Chen, 1996). In the approach, however, a sentence which is not accepted by the grammar is assigned zero probability. Thus, the grammar must have broad-coverage so that any sentence will get non-zero probability. But acquisition of such a robust grammar has been known to be very difficult. Due to the difficulty, some works try to use an integrated model of grammar and n-gram compensating each other(McCandless, 1994; Meteer and Rohlicek, 1993). Given a robust grammar, grammar-based language modeling is expected to be more powerful and compact in model size than n-gram-based one.

In this paper we present a language modeling based on a kind of simple dependency grammar. The grammar consists of head-dependent relations between words and can be learned automatically from a raw corpus using the reestimation algorithm which is also introduced in this paper. Based on the dependencies, a sentence is analyzed and assigned syntactic structures by which long distance dependences are represented. Because the model can be thought of as a linguistic bi-gram model, the smoothing functions of n-gram models can be applied to it. Thus, the model can be robust, adapt easily to new domains, and be effective.

The paper is organized as follows. We introduce some definitions and notations for the dependency grammar and the reestimation algorithm in section 2, and explain the algorithm in section 3. In section 4, we show the experimental results for the suggested model compared to n-gram models. Finally, section 5 concludes this paper.

2 A Simple Dependency Grammar

In this paper, we assume a kind of simple dependency grammar which describes a language

by a set of head-dependent relations between words. A sentence is analyzed by establishing dependency links between individual words in the sentence. A dependency analysis, $\mathcal{D}$, of a sentence can be represented with arrows pointing from head to dependent as depicted in Figure 1. For structural generality, we assume that there is always a marking tag, "EOS"(End of Sentence), at the end of a sentence and it has the head word of the sentence as its own dependent("gave" in Figure 1).

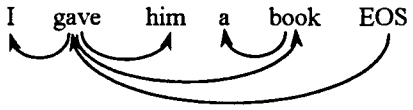

Figure 1: An example dependency analysis

A $\mathcal{D}$ is a set of inter-word dependencies which satisfy the following conditions: (1) every word in the sentence has its head in the sentence except the head word of the sentence. (2) every word can have only one head. (3) there is neither crossing nor cycle of dependencies.

The probabilistic model of the simple dependency grammar is given by

$$p(sentence) = \sum_{\mathcal{D}} p(\mathcal{D}) = \sum_{\mathcal{D}} \prod_{x \to y \in \mathcal{D}} p(x \to y),$$

where

$$p(x \to y) = p(y|x) = \frac{freq(x \to y)}{\sum_z freq(x \to z)}.$$

Complete-Link and Complete-Sequence

Here, we define complete-link and complete-sequence which represent partial $\mathcal{D}$s for substrings. They are used to construct overall $\mathcal{D}$s and used as the basic structures for the reestimation algorithm in section 3.

A set of dependency relations on a word sequence, $w_{i,j}$[1], is a complete-link when the following conditions are satisfied:

- there is $(w_i \to w_j)$ or $(w_i \leftarrow w_j)$ exclusively.
- Every inner word has a head in the word sequence.
- Neither crossing nor cycle of dependency relations is allowed.

[1]We use w_i for ith word in a sentence and $w_{i,j}$ for the word sequence from w_i to $w_j (i < j)$.

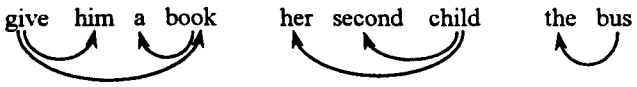

Figure 2: Example complete-links

A complete-link has direction. A complete-link on $w_{i,j}$ is said to be "rightward" if the outermost relation is $(w_i \to w_j)$, and "leftward" if the relation is $(w_i \leftarrow w_j)$. Unit complete-link is defined on a string of two adjacent words, $w_{i,i+1}$. In Figure 2, (a) is a rightward complete-link, and both of (b) and (c) are leftward ones.

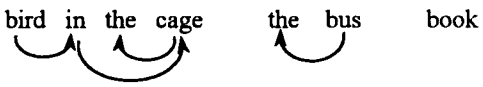

Figure 3: Example complete-sequences

A complete-sequence is a sequence of 0 or more adjacent complete-links that have the same direction. A unit complete-sequence is defined on a string of one word. It is 0 sequence of complete-links. The direction of a complete-sequence is determined by the direction of the component complete-links. In Figure 3, (a) is a rightward complete-sequence composed of two complete-links, and (b) is a leftward one. (c) is a complete-sequence composed of zero complete-links, and it can be both leftward and rightward.

The word of "complete" means that the dependency relations on the inner words are completed and that consequently there is no need to process further on them. From now on, we use $L_r(i,j)/L_l(i,j)$ for rightward/leftward complete-links and $S_r(i,j)/S_l(i,j)$ for rightward/leftward complete-sequences on $w_{i,j}$.

Any complete-link on $w_{i,j}$ can be viewed as the following combination.

- $L_r(i,j)$: $\{(w_i \to w_j), S_r(i,m), S_l(m+1,j)\}$
- $L_l(i,j)$: $\{(w_i \leftarrow w_j), S_r(i,m), S_l(m+1,j)\}$ for a $m (i \leq m < j)$.

Otherwise, the set of dependencies does not satisfy the conditions of no crossing, no cycle and no multiple heads and is not a complete-link any more.

Similarly, any complete-sequence on $w_{i,j}$ can be viewed as the following combination.

- $S_r(i,j)$: $\{S_r(i,m), L_r(m,j)\}$
- $S_l(i,j)$: $\{L_l(i,m), S_l(m,j)\}$ for a $m (i \leq m < j)$.

In the case of complete-sequence, we can prevent multiple constructions of the same

complete-sequence by the above combinational restriction.

Figure 4: Abstract representation of $\mathcal{D}$

Figure 4 shows an abstract representation of a $\mathcal{D}$ of an n-word sentence. When $w_k (1 \leq k \leq n)$ is the head of the sentence, any $\mathcal{D}$ of the sentence can be represented by a $S_l(1, EOS)$ uniquely by the assumption that there is always the dependency relation, $(w_k \leftarrow w_{EOS})$.

3 Reestimation Algorithm

The reestimation algorithm is a variation of Inside-Outside algorithm(Jelinek et al., 1990) adapted to dependency grammar. In this section we first define the inside-outside probabilities of complete-links and complete-sequences, and then describe the reestimation algorithm based on them[2].

In the followings, β indicates inside probability and α is for outside probability. The superscripts, l and s, are used for "complete-link" and "complete-sequence" respectively. The subscripts indicate direction: r for "rightward" and l for "leftward".

The inside probabilities of complete-links $(L_r(i,j),\ L_l(i,j))$ and complete-sequences $(S_r(i,j),\ S_l(i,j))$ are as follows.

$$\beta_r^l(i,j) = \sum_{m=i}^{j-1} p(w_i \rightarrow w_j)\beta_r^s(i,m)\beta_l^s(m+1,j).$$

$$\beta_l^l(i,j) = \sum_{m=i}^{j-1} p(w_i \leftarrow w_j)\beta_r^s(i,m)\beta_l^s(m+1,j).$$

$$\beta_r^s(i,j) = \sum_{m=i}^{j-1} \beta_r^s(i,m)\beta_r^l(m,j).$$

$$\beta_l^s(i,j) = \sum_{m=i+1}^{j} \beta_l^l(i,m)\beta_l^s(m,j).$$

The basis probabilities are:

$$\beta_r^l(i,i+1) = p(w_i \rightarrow w_{i+1})$$
$$\beta_l^l(i,i+1) = p(w_i \leftarrow w_{i+1})$$
$$\beta_r^s(i,i) = \beta_l^s(i,i) = 1$$
$$\beta_l^s(1,EOS) = p(w_{1,n})$$
$$\beta_r^s(i,i+1) = p(L_r(i,i+1)) = p(w_i \rightarrow w_{i+1})$$
$$\beta_l^s(i,i+1) = p(L_l(i,i+1)) = p(w_i \leftarrow w_{i+1}).$$

$\beta_l^s(1,EOS)$ is the sentence probability because every dependency analysis, $\mathcal{D}$, is represented by a $S_l(1,EOS)$ and $\beta_l^s(1,EOS)$ is sum of the probability of every $S_l(1,EOS)$.

The outside probabilities for complete-links $(L_r(i,j),\ L_l(i,j))$ and complete-sequences $(S_r(i,j),\ S_l(i,j))$ are as follows.

$$\alpha_r^l(i,j) = \sum_{v=1}^{i} \alpha_r^s(v,j)\beta_r^s(v,i).$$

$$\alpha_l^l(i,j) = \sum_{h=j}^{n} \alpha_l^s(i,h)\beta_l^s(j,h).$$

$$\begin{aligned}\alpha_r^s(i,j) = &\sum_{h=j+1}^{n} \alpha_r^s(i,h)\beta_r^l(j,h) \\ &+\alpha_r^l(i,h)\beta_l^s(j+1,h)p(w_i \rightarrow w_h) \\ &+\alpha_l^l(i,h)\beta_l^s(j+1,h)p(w_i \leftarrow w_h).\end{aligned}$$

$$\begin{aligned}\alpha_l^s(i,j) = &\sum_{v=1}^{i-1} \alpha_l^s(v,j)\beta_l^l(v,i) \\ &+\alpha_r^l(v,j)\beta_r^s(v,i-1)p(w_v \rightarrow w_j) \\ &+\alpha_l^l(v,j)\beta_r^s(v,i-1)p(w_v \leftarrow w_j).\end{aligned}$$

The basis probability is

$$\alpha_l^s(1,EOS) = 1.$$

Given a training corpus, the initial grammar is just a list of all pairs of unique words in the corpus. The initial pairs represent the tentative head-dependent relations of the words. And the initial probabilities of the pairs can be given randomly. The training starts with the initial grammar. The train corpus is analyzed with the grammar and the occurrence frequency of each dependency relation is calculated. Based on the frequencies, probabilities of dependency relations are recalculated by $p_e(w_p \rightarrow w_q) = \frac{C(w_p \rightarrow w_q)}{\sum_{w_r} C(w_p \rightarrow w_r)}$. The process continues until the entropy of the training corpus becomes the minimum. The frequency of occurrence, $C(w_i \rightarrow w_j)$, is calculated by

$$\begin{aligned}C(w_i \rightarrow w_j) &= \sum_{\mathcal{D}} p(\mathcal{D}|w_{1,n})O_{cc}(w_i \rightarrow w_j, \mathcal{D}, w_{1,n}) \\ &= \frac{1}{p(w_{1,n})}\alpha_r^l(i,j)\beta_r^l(i,j)\end{aligned}$$

where $O_{cc}(w_i \rightarrow w_j, \mathcal{D}, w_{1,n})$ is 1 if the dependency relation, $(w_i \rightarrow w_j)$, is used in the $\mathcal{D}$,

[2] A little more detailed explanation of the expressions can be found in (Lee and Choi, 1997).

and 0 otherwise. Similarly, the occurrence frequency of the dependency relation, $(w_i \leftarrow w_j)$, is computed by $\frac{1}{p(w_{1,n})}\alpha_l^l(i,j)\beta_l^l(i,j)$.

4 Preliminary experiments

We have experimented with three language models, tri-gram model (TRI), bi-gram model (BI), and the proposed model (DEP) on a raw corpus extracted from KAIST corpus[3]. The raw corpus consists of 1,589 sentences with 13,139 words, describing animal life in nature. We randomly divided the corpus into two parts: a training set of 1,445 sentences and a test set of 144 sentences. And we made 15 partial training sets which include the first s sentences in the whole training set, for s ranging from 100 to 1,445 sentences. We trained the three language models for each partial training set, and tested the training and the test corpus entropies.

TRI and BI was trained by counting the occurrence of tri-grams and bi-grams respectively. DEP was trained by running the reestimation algorithm iteratively until it converges to an optimal dependency grammar. On the average, 26 iterations were done for the training sets.

Smoothing is needed for language modeling due to the sparse data problem. It is to compensate for the overestimated and the underestimated probabilities. Smoothing method itself is an important factor. But our goal is not to find out a better smoothing method. So we fixed on an interpolation method and applied it for the three models. It can be represented as (McCandless, 1994)

$$\hat{P}_n(w_i|w_{i-n+1},...,w_{i-1}) = \lambda P_n(w_i|w_{i-n+1},...,w_{i-1}) + (1-\lambda)\hat{P}_{n-1}(w_i|w_{i-n+2},...,w_{i-1}),$$

where

$$\lambda = \frac{C(w_1,...,w_{n-1})}{C(w_1,...,w_{n-1}) + K_s}.$$

The K_s is the global smoothing factor. The bigger the K_s, the larger the degree of smoothing. For the experiments we used 2 for K_s.

We take the performance of a language model to be its cross-entropy on test corpus,

$$\frac{1}{|V|}\sum_{i=1}^{S} -log_2 p_m(s_i)$$

where the test corpus contains a total of $|V|$ words and is composed of S sentences.

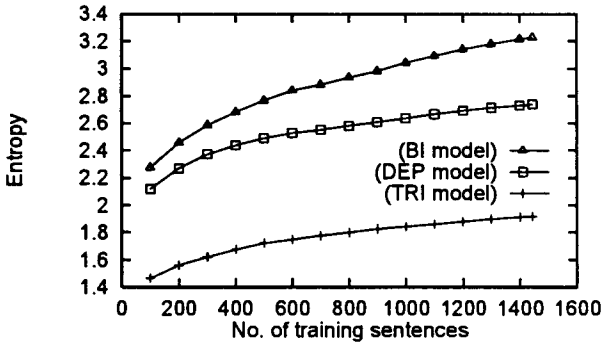

Figure 5: Training corpus entropies

Figure 5 shows the training corpus entropies of the three models. It is not surprising that DEP performs better than BI. DEP can be thought of as a kind of linguistic bi-gram model in which long distance dependencies can be represented through the head-dependent relations between words. TRI shows better performance than both BI and DEP. We think it is because TRI overfits the training corpus, judging from the experimental results for the test corpus.

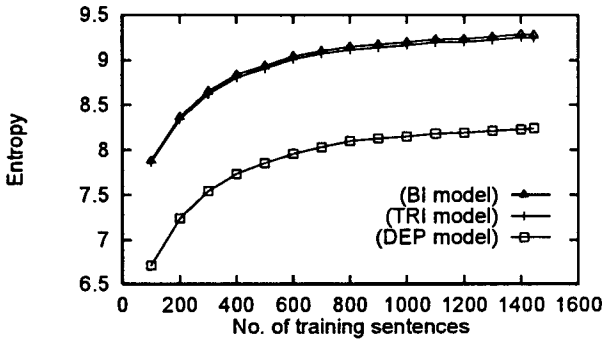

Figure 6: Test corpus entropies

For the test corpus, BI shows slightly better performance than TRI as depicted in Figure 6. Increase in the order of n-gram from two to three shows no gains in entropy reduction. DEP, however, shows still better performance than the n-gram models. It shows about 11.5% entropy reduction to BI and about 11% entropy reduction to TRI. Figure 7 shows the entropies for the mixed corpus of training and test sets. From the results, we can see that head-dependent relations between words are more useful information than the naive n-gram sequences, for language modeling. We can see also that the reestimation algorithm can find out properly the hidden head-dependent relations between words, from a raw corpus.

[3]KAIST (Korean Advanced Institute of Science and Technology) corpus has been under construction since 1994. It consists of raw text collection(45,000,000 words), POS-tagged collection(6,750,000 words), and tree-tagged collection(30,000 sentences) at present.

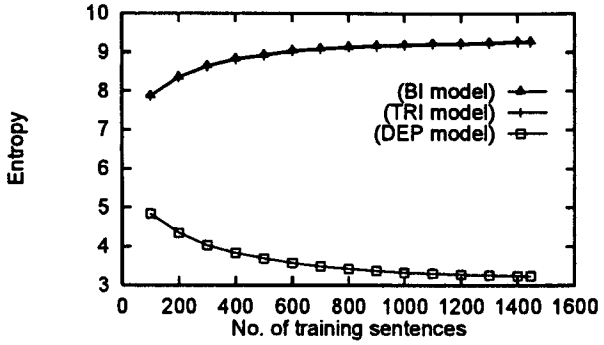

Figure 7: Mixed corpus entropies

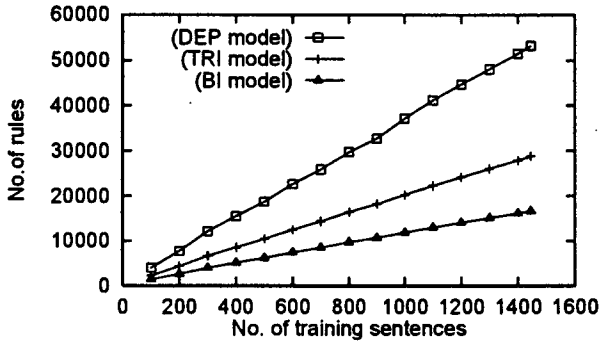

Figure 8: Model size

Related to the size of model, however, DEP has much more parameters than TRI and BI as depicted in Figure 8. This can be a serious problem when we create a language model from a large body of text. In the experiments, however, DEP used the grammar acquired automatically as it is. In the grammar, many inter-word dependencies have probabilities near 0. If we exclude such dependencies as was experimented for n-grams by Seymore and Rosenfeld (1996), we may get much more compact DEP model with very slight increase in entropy.

5 Conclusions

In this paper, we presented a language model based on a kind of simple dependency grammar. The grammar consists of head-dependent relations between words and can be learned automatically from a raw corpus by the reestimation algorithm which is also introduced in this paper. By the preliminary experiments, it was shown that the proposed language model performs better than n-gram models in test corpus entropy. This means that the reestimation algorithm can find out the hidden information of head-dependent relation between words in a raw corpus, and the information is more useful than the naive word sequences of n-gram, for language modeling.

We are planning to experiment the performance of the proposed language model for large corpus, for various domains, and with various smoothing methods. For the size of the model, we are planning to test the effects of excluding the dependency relations with near zero probabilities.

References

P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. 1992. "Class-Based n-gram Models of Natural Language". *Computational Linguistics*, 18(4):467–480.

C. Chang and C. Chen. 1996. "Application Issues of SA-class Bigram Language Models". *Computer Processing of Oriental Languages*, 10(1):1–15.

S. F. Chen. 1996. *"Building Probabilistic Models for Natural Language"*. Ph.D. thesis, Havard University, Cambridge, Massachusetts.

F. Jelinek, J. D. Lafferty, and R. L. Mercer. 1990. "Basic Methods of Probabilistic Context Free Grammars". Technical report, IBM – T.J. Watson Research Center.

K. Lari and S. J. Young. 1991. "Applications of stochastic context-free grammars using the inside-outside algorithm". *Computer Speech and Language*, 5:237–257.

S. Lee and K. Choi. 1997. "Reestimation and Best-First Parsing Algorithm for Probabilistic Dependency Grammar". In *WVLC-5*, pages 11–21.

M. K. McCandless. 1994. "Automatic Acquisition of Language Models for Speech Recognition". Master's thesis, Massachusetts Institute of Technology.

M. Meteer and J.R. Rohlicek. 1993. "Statistical Language Modeling Combining N-gram and Context-free Grammars". In *ICASSP-93*, volume II, pages 37–40, January.

K. Seymore and R. Rosenfeld. 1996. "Scalable Trigram Backoff Language Models". Technical Report CMU-CS-96-139, Carnegie Mellon University.

S. Sneff. 1992. "TINA: A natural language system for spoken language applications". *Computational Linguistics*, 18(1):61–86.