Dependency Structure Applied to Language Modeling for Information Retrieval

Changki Lee, Gary Geunbae Lee, and Myung-Gil Jang

In this paper, we propose a new language model, namely, a dependency structure language model, for information retrieval to compensate for the weaknesses of unigram and bigram language models. The dependency structure language model is based on the first-order dependency model and the dependency parse tree generated by a linguistic parser. So, long-distance dependencies can be naturally captured by the dependency structure language model. We carried out extensive experiments to verify the proposed model, where the dependency structure model gives a better performance than recently proposed language models and the Okapi BM25 method, and the dependency structure is more effective than unigram and bigram in language modeling for information retrieval.

Keywords: Language model, term dependency, information retrieval, dependency structure.

Manuscript received Jan. 31, 2005; revised Apr. 17, 2006.

The material in this work was in part presented at ACM SIGIR 2003 MF/IR workshop.

Changki Lee (phone: + 82 42 860 6879, email: leeck@etri.re.kr) and Myung-Gil Jang (email: mgjang@etri.re.kr) are with Embedded Software Research Division, ETRI, Daejeon, Korea.

Gary Geunbae Lee (email: gblee@postech.ac.kr) is with the Department of Computer Science & Engineering, POSTECH, Pohang, Gyeongbuk, Korea.

I. Introduction

Using language models for information retrieval has recently been studied extensively [1]-[4]. The basic idea is to compute the conditional probability $p(q|d)$, that is, the probability of generating a query q given the observation of a document d. Several different methods have been applied to compute this conditional probability.

Ponte and Croft [2] used several heuristics to smooth the maximum likelihood estimate (MLE) of the document language model, and assumed that the query is generated under a multivariate Bernoulli model. The BBN method [1] uses a two-state hidden Markov model as the basis for generating queries, which in effect is to smooth the MLE with linear interpolation. In Zhai and Lafferty [4], it has been found that the retrieval performance is affected by both the estimation accuracy of document language models and the appropriate modeling of the query.

The language models used in most previous works are the unigram models (similarly, Ponte and Croft [2] assume that, given a particular language model, the query terms occur independently). The unigram language model makes a strong assumption that each word occurs independently, and consequently, the probability of a word sequence becomes the product of the probabilities of the individual words.

There are some explorations of bigram and trigram models to improve this unrealistic assumption by considering the local context [1], [3]. For a bigram, the probability of a new word depends on the previous word, while for a trigram, the probability of a new word depends on the probabilities of the previous two words.

However, bigram and trigram models have a limitation in handling long-distance dependences (in the speech literature, the term `long-distance dependencies' regularly refers to anything beyond the range of a trigram model). In a question such as

"Which book should Peter read?"

we can recognize a long distance dependency between 'read' and 'book'. Recently, there have been language models based on syntactic parsing to overcome the limitation of bigrams/trigrams in the speech recognition field [5]-[7].

Van Rijsbergen explored one way of removing the independence assumption using the Chow Expansion theory in information retrieval [8]. He constructed a probabilistic model incorporating dependences between index terms using a maximum spanning tree approach. The extent to which two index terms depend on each other is derived from the distribution of co-occurrences in the entire collection or in the relevant and non-relevant document sets.

Jianfeng Gao and others proposed a dependence language modeling approach to information retrieval [9]. The approach extends the basic language modeling approach based on unigram by relaxing the independence assumption. They generated a term co-occurrence model, where any term pair within a term trigram in a sentence has a link. But they did not use a linguistic syntactic structure.

In this paper, which addresses similar concepts, we propose a dependency structure language model to overcome the limitation of unigram and bigram models in information retrieval. The dependency structure language model is based on a dependency parse tree generated by linguistic parser. So, long-distance dependencies can be naturally handled by the linguistic syntactic structure model. Our dependency structure language model adopts the first-order dependency model and dependency parse tree to capture long-distance dependencies in information retrieval applications.

The remainder of this paper is organized as follows. In section II, we describe the first-order dependency model, while in section III, we describe the dependency structure language model. In section IV, we present some experiments and their results. Section V gives our conclusion and future work.

II. First-Order Dependency Model

Consider query $q = q_1, q_2, \cdots, q_n$ and the corresponding vector $x = \{x_1, x_2, \cdots, x_n\}$, where $x_i = 1$ if q_i appears in a document, and $x_i = 0$ otherwise. The problem of estimating a density becomes the problem of estimating the probability $p(x)$. Since there are 2^n possible vectors x, we must estimate 2^n probabilities, which is an enormous task.

If the components of x are statistically independent, the problem is greatly simplified. In this case we can write

$$p(x) = \prod_{i=1}^{n} p(x_i) = \prod_{i=1}^{n} p_i^{x_i}(1-p_i)^{1-x_i} \ ,$$

where $p_i = p(x_i=1)$ and $1-p_i = p(x_i=0)$.

It is natural to ask whether or not there are any compromise positions between being completely accurate, which requires estimating 2^n probabilities, and being forced to assume statistical independence, which reduce the problem to estimate only n probabilities. One answer is provided by finding an expansion for $p(x)$ and approximating $p(x)$ by a partial sum, for example, the Rademacher-Walsh Expansion and the Bahadur-Lazarsfeld Expansion [10]. Another interesting class of approximation to a joint probability distribution $p(x)$ is based on the identity

$$\begin{aligned} p(x) &= p(x_1, x_2, x_3, \cdots, x_n) \\ &= p(x_1)p(x_2|x_1)\ldots p(x_n|x_{n-1}, x_{n-2}, \cdots, x_1). \end{aligned} \tag{1}$$

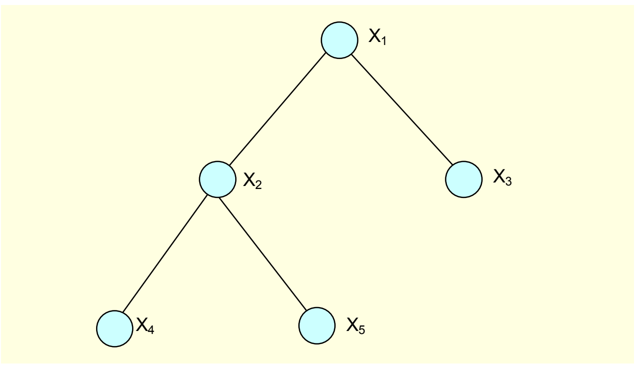

Fig. 1. A dependence tree.

Suppose the variables are not independent, but we can number the variables so that $p(x_i|x_{i-1}, \cdots, x_1)$ is solely dependent on the preceding variable $x_{j(i)}$. For example, suppose that

$$\begin{aligned} p(x_5|x_4,x_3,x_2,x_1) &= p(x_5|x_{j(5)}) = p(x_5|x_2) \\ p(x_4|x_3,x_2,x_1) &= p(x_4|x_{j(4)}) = p(x_4|x_2) \\ p(x_3|x_2,x_1) &= p(x_3|x_{j(3)}) = p(x_3|x_1) \\ p(x_2|x_1) &= p(x_2|x_{j(2)}) = p(x_2|x_1) \end{aligned}$$

with a corresponding dependence tree as in Fig. 1. It then follows from (1) that $p(x_1,x_2,x_3,x_4,x_5)$ can be written as $p(x_1)p(x_2 | x_1)p(x_3 | x_1)p(x_4 | x_2)p(x_5 | x_2)$.

Chow and Liu suggest the construction of a tree such that the mutual information between a variable and the variable immediately above it are maximized for a dependence tree, as in Fig. 1, which was originally used in the Chow Expansion [11]. Given two points on the tree such that the i-th point is directly and immediately above the j-th point, a maximum spanning tree (MST) may be defined as maximizing the sum:

$$\sum_{i,j} I(i, j),$$

where $I(i, j)$ represents the expected mutual information provided by i about j,

$$I(i, j) = \sum_{i,j} p(q_i, q_j) \log \frac{p(q_i, q_j)}{p(q_i)p(q_j)}.$$

A dependency relationship [12] is an asymmetric binary relationship between a *head* word (or governor, parent), and *modifier* word (or dependent, daughter). Dependency grammars represent sentence structures as a set of dependency relationships. Normally, the dependency relationships from a tree connect all the words in a sentence. A word in the sentence may have several modifiers, but each word may modify at most one word. The root of the dependency tree does not modify any word. It is also called the head of the sentence.

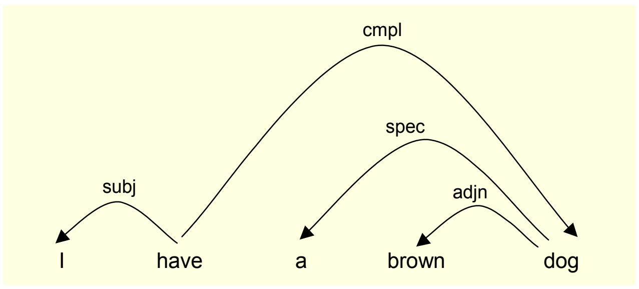

Fig. 2. A dependency structure of a sentence.

For example, Fig. 2 is a dependency structure of a sentence. The head of the sentence is ‘have’. There are four pairs of dependency relationships, depicted by four arcs from the heads to the modifiers.

We use Minipar as a dependency parser, which is a principle-based English parser [13]. Minipar represents its grammar as a network where nodes represent grammatical categories and links represent types of dependency relationships. An evaluation with the SUSANNE corpus shows that Minipar achieves about 88% precision and 80% recall with respect to dependency relationships. Minipar is one of the more efficient parsers. It parses about 300 words per second on a Pentium II 300 with 128 MB memory [14], yet it would be difficult to run experiments on larger collections even with the fastest of parsers because of the speed issue.

Chow and Liu suggest the construction of an MST using mutual information for a dependence tree, which was originally used in the Chow Expansion. However, we suggest using a dependency parse tree that is generated by a linguistic dependency parser instead of the mutual information MST because a dependency parse tree intuitively and linguistically represents the term dependence relations in the syntactic structure, which helps to capture the underlying semantics of a document.

III. Dependency Structure Language Model

The idea of the language modeling approach to information retrieval is to estimate the language model for a document and then to compute the likelihood that the query would have been generated from the estimated model.

Given query q and document d, we are interested in estimating the conditional probability $p(d|q)$, that is, the probability that d fits the observed q. After applying the Bayes' formula and dropping a document-independent constant, we have $p(q|d)p(d)$. Here, $p(d)$ is a prior belief that d is relevant to any query and $p(q|d)$ is the query likelihood given the document, which captures how well the document generates the particular query q.

In the simplest case, $p(d)$ is assumed to be uniform, and so does not affect document ranking. This assumption has been taken in most previous works. In our study, we also assume a uniform $p(d)$ in order to focus on the effect of dependency structure. With the prior uniformity, the retrieval model reduces to the calculation of $p(q|d)$, where language modeling comes in.

The language models used in most previous works are the unigram models, which are the multinomial models that assign the probability,

$$p(q|d) = \prod_i p(q_i|d).$$

Clearly, the retrieval problem is now essentially reduced to unigram language model estimation. The unigram language model makes a strong assumption that each word occurs independently, and consequently, the probability of a word sequence becomes the product of the probabilities of the individual words. However, the unigram model has some limitations to capture the term relations in a document.

The basic idea of our dependency structure language model is to capture the term relations in a linguistically practical way and can be described as follows. An interesting approximation to a joint probability distribution $p(q|d)$ is based on the identity

$$\begin{aligned} p(q|d) &= p(q_1, q_2, \ldots, q_n|d) \\ &= p(q_1|d)\, p(q_2|q_1, d) \cdots p(q_n|q_{n-1}, \cdots, q_1, d). \end{aligned}$$

We can number the words so that $p(q_i \mid q_{i-1},\cdots,q_1,d)$ is solely dependent on some preceding word $q_{j(i)}$ as in the Chow Expansion theory. We then obtain the product expansion

$$p(q|d) = p(q_1|d)\, p(q_2|q_{j(2)}, d) \ldots p(q_n|q_{j(n)}, d) \qquad (2)$$

The function $j(i)$ is obtained from the dependency parse tree of a query. We assume the query is a sentence or at least a fragment of sentence whose Minipar result is reasonable. Thus, if the query is simply a bag of words, the Minipar result might be harmful for the performance. And it is impossible to use the count of a query term to indicate the importance of the term as in a keyword search scenario. Furthermore, it is hard to deal with the case when a query term occurs more than once in the same sentence. For example, assume the query is "how to search with the search engine?" Thus, the term 'search' corresponds to two nodes in the dependence tree. If the term 'search' occurs in a document, it is unclear which node in the dependence tree we should associate the term with. In this case, we assume the term is associated with the node that occurred first.

By letting $x_i = 1$ if q_i appears in document d, and $x_i = 0$ otherwise, we can write the probability of q_i given $q_{j(i)}$ as follows:

$$p(q_i|q_{j(i)}, d) = (p_s(q_i|q_{j(i)}, d)^{x_i} p_u(q_i|q_{j(i)}, d)^{1-x_i})^{x_{j(i)}} \times (p_s(q_i|d)^{x_i} p_u(q_i|d)^{1-x_i})^{1-x_{j(i)}}, \quad (3)$$

where $p_s(q_i|q_{j(i)}, d) = p(x_i=1|x_{j(i)}=1, d)$, $p_u(q_i|q_{j(i)}, d) = p(x_i=0|x_{j(i)}=1, d)$, $p_s(q_i|d) = p(x_i=1|x_{j(i)}=0, d) \approx p(x_i=1|d)$, and $p_u(q_i|d) = p(x_i=0|x_{j(i)}=0, d) \approx p(x_i=0|d)$. In the equation, $p_s(q_i|d)$ is used for 'seen' word q_i that occurs in document d, and $p_u(q_i|d)$ for 'unseen' word q_i that does not. Probability $p_s(q_i|q_{j(i)}, d)$ is used when q_i and $q_{j(i)}$ occur in document d, where they occur as a dependency relation. Probability $p_u(q_i|q_{j(i)}, d)$ is used when q_i and $q_{j(i)}$ occur in document d, but without any dependency relation between them.

By substituting (3) into (2), taking the logarithm, and collecting the terms, we obtain the following equation as in the Chow Expansion theory.

$$\begin{aligned}
&\log p(q|d) \\
&= \log p(q_1|d) + \sum_{i=2}^{n} \log p(q_i|q_{j(i)}, d) \\
&= \log(p_s(q_1|d)^{x_1} p_u(q_1|d)^{1-x_1}) \\
&\quad + \sum_{i=2}^{n} x_{j(i)} \log(p_s(q_i|q_{j(i)}, d)^{x_i} p_u(q_i|q_{j(i)}, d)^{1-x_i}) \\
&\quad + \sum_{i=2}^{n} (1 - x_{j(i)}) \log(p_s(q_i|d)^{x_i} p_u(q_i|d)^{1-x_i}) \\
&= \sum_{i=1}^{n} x_i \log \frac{p_s(q_i|d)}{p_u(q_i|d)} + \sum_{i=2}^{n} x_{j(i)} \log \frac{p_u(q_i|q_{j(i)}, d)}{p_u(q_i|d)} \\
&\quad + \sum_{i=2}^{n} x_i x_{j(i)} \left(\log \frac{p_s(q_i|q_{j(i)}, d)}{p_u(q_i|q_{j(i)}, d)} - \log \frac{p_s(q_i|d)}{p_u(q_i|d)} \right) + \sum_{i=1}^{n} \log p_u(q_i|d)
\end{aligned} \quad (4)$$

Let $c(q_i;d)$ denote the count of word q_i in document d, and $c(q_i,q_{j(i)};d)$ denote the count of occurrence of q_i and $q_{j(i)}$ as a dependency relation in document d. Then, $x_i = 1$ means $c(q_i;d) > 0$, $x_{j(i)} = 1$ means $c(q_{j(i)};d) > 0$, and $x_i x_{j(i)} = 1$ means $c(q_i,q_{j(i)};d) > 0$. So, we can re-write (4) in $c(q_i;d)$ and $c(q_i,q_{j(i)};d)$ terms as follows:

$$\begin{aligned}
\log p(q|d) = &\sum_{i:c(q_i;d)>0} \log \frac{p_s(q_i|d)}{p_u(q_i|d)} + \sum_{i:c(q_{j(i)};d)>0} \log \frac{p_u(q_i|q_{j(i)}, d)}{p_u(q_i|d)} \\
&+ \sum_{c(q_i,q_{j(i)};d)>0} \left(\log \frac{p_s(q_i|q_{j(i)}, d)}{p_u(q_i|q_{j(i)}, d)} - \log \frac{p_s(q_i|d)}{p_u(q_i|d)} \right) \\
&+ \sum_{i} \log p_u(q_i|d).
\end{aligned} \quad (5)$$

Now, we can see that the retrieval function can actually be decomposed into four parts. The first part involves a weight for each term, which is common between the query and the document (that is, matched terms). The second part involves a weight for head (or governor, parent) terms of matched terms. The third part involves a weight for the occurrence of matched terms and their head terms as a dependency relation. The last part only involves a document-dependent constant that is related to how much probability mass will be allocated to unseen words according to the particular smoothing method used.

In Zhai and Lafferty [4], three smoothing methods, Jelinek-Mercer, Dirichlet, and absolute discounting are compared. In the comparison, Jelinek-Mercer and Dirichlet clearly have a better average precision than absolute discounting. Considering these results, we use two smoothing methods, Jelinek-Mercer and Dirichlet, for our dependency structure language model.

1. Jelinek-Mercer Smoothing Method

This method involves a linear interpolation of the maximum likelihood model with the fallback model (that is, collection model), using a coefficient λ to control the influence of each model:

$$p_\lambda(q_i|d) = (1 - \lambda) \cdot p_{ml}(q_i|d) + \lambda \cdot p(q_i|C).$$

Using this smoothing method, we define $p_s(q_i|d)$ and $p_u(q_i|d)$ as follows:

$$\begin{aligned}
p_s(q_i|d) &= (1 - \lambda_1) \cdot p_{ml}(q_i|d) + \lambda_1 \cdot p(q_i|C), \\
p_u(q_i|d) &= \lambda_1 \cdot p(q_i|C),
\end{aligned} \quad (6)$$

where $p_{ml}(q_i|d) = [(c(q_i;d))/(\sum_k c(q_k;d))]$.

We also define $p_s(q_i|q_{j(i)}, d)$ and $p_u(q_i|q_{j(i)}, d)$ using the same smoothing method as follows:

$$p_s(q_i|q_{j(i)},d)=(1-\lambda_2)\cdot p_{ml}(q_i|q_{j(i)},d)+\lambda_2\cdot p(q_i|q_{j(i)},C),$$
$$p_u(q_i|q_{j(i)},d)=\lambda_2\cdot p(q_i|q_{j(i)},C), \tag{7}$$

where $p_{ml}(q_i|q_{j(i)},d) = [(c(q_i,q_{j(i)};d))/(\sum_k c(q_k,q_{j(i)};d))]$.

By substituting (6) and (7) into (5), we obtain the following equation.

$$\begin{aligned}\log p(q|d) =& \sum_{i:c(q_i;d)>0} \log\left(1+\frac{(1-\lambda_1)\cdot p_{ml}(q_i|d)}{\lambda_1\cdot p(q_i|C)}\right)\\ &+\sum_{i:c(q_{j(i)};d)>0} \log\frac{\lambda_2\cdot p(q_i|q_{j(i)},C)}{\lambda_1\cdot p(q_i|C)}\\ &+\sum_{i:c(q_i,q_{j(i)};d)>0} \log\left(1+\frac{(1-\lambda_2)\cdot p_{ml}(q_i|q_{j(i)},d)}{\lambda_2\cdot p(q_i|q_{j(i)},C)}\right)\\ &-\sum_{i:c(q_i,q_{j(i)};d)>0} \log\left(1+\frac{(1-\lambda_1)\cdot p_{ml}(q_i|d)}{\lambda_1\cdot p(q_i|C)}\right)\\ &+\sum_i \log(\lambda_1\cdot p(q_i|C))\end{aligned} \tag{8}$$

In (8), we ignore the last part, the document-independent constant, in order to focus on the effects of the second, third, and fourth parts.

From (8), we define $MS_{DSLM\text{-}J}(q,d)$, a query-document scoring function adapted from the dependency structure language model using the Jelinek-Mercer smoothing method, as follows:

$$\begin{aligned}MS_{DSLM-J}(q,d) =& \sum_{i:c(q_i;d)>0} \log\left(1+\frac{(1-\lambda_1)\cdot p_{ml}(q_i|d)}{\lambda_1\cdot p(q_i|C)}\right)\\ &+\sum_{i:c(q_{j(i)};d)>0} k\cdot\log\frac{\lambda_2\cdot p(q_i|q_{j(i)},C)}{\lambda_1\cdot p(q_i|C)}\\ &+\sum_{i:c(q_i,q_{j(i)};d)>0} k\cdot\log\left(1+\frac{(1-\lambda_2)\cdot p_{ml}(q_i|q_{j(i)},d)}{\lambda_2\cdot p(q_i|q_{j(i)},C)}\right)\\ &-\sum_{i:c(q_i,q_{j(i)};d)>0} k\cdot\log\left(1+\frac{(1-\lambda_1)\cdot p_{ml}(q_i|d)}{\lambda_1\cdot p(q_i|C)}\right),\end{aligned} \tag{9}$$

where k is a constant parameter to control the influence of each part. Thus if $k = 0$ gives a unigram language model and $k = 1$ gives a fully dependency structured language model, different values of k give a mix of the two. In the formula, $p(q_i|q_{j(i)}, C)$ can be zero because of a data sparseness problem. To solve this problem, we also apply the same smoothing to $p(q_i|q_{j(i)}, C)$ as follows:

$$p(q_i|q_{j(i)}, C) = (1-\lambda_3)\cdot p_{ml}(q_i|q_{j(i)}, C) + \lambda_3\cdot p(q_i|C).$$

2. Dirichlet Smoothing Method

A language model is a multinomial distribution, for which the conjugate prior for Bayesian analysis is the Dirichlet distribution with parameters

$$(\mu\cdot p(q_1|C), \mu\cdot p(q_2|C),\cdots, \mu\cdot p(q_n|C)).$$

Thus, the model is given by

$$p_\mu(q_i|d)=\frac{c(q_i;d)+\mu\cdot p(q_i|C)}{\sum_k c(q_k;d)+\mu}.$$

Using this smoothing method, we define $p_s(q_i|d)$ and $p_u(q_i|d)$ as follows:

$$p_s(q_i|d)=\frac{c(q_i;d)+\mu_1\cdot p(q_i|C)}{\sum_k c(q_k;d)+\mu_1},$$
$$p_u(q_i|d)=\frac{\mu_1\cdot p(q_i|C)}{\sum_k c(q_k;d)+\mu_1}. \tag{10}$$

We also define $p_s(q_i|q_{j(i)}, d)$ and $p_u(q_i|q_{j(i)}, d)$ using the same smoothing method as follows:

$$p_s(q_i|q_{j(i)},d)=\frac{c(q_i,q_{j(i)};d)+\mu_2\cdot p(q_i|q_{j(i)},C)}{\sum_k c(q_k,q_{j(i)};d)+\mu_2},$$
$$p_u(q_i|q_{j(i)},d)=\frac{\mu_2\cdot p(q_i|q_{j(i)},C)}{\sum_k c(q_k,q_{j(i)};d)+\mu_2}. \tag{11}$$

By substituting (10) and (11) into (5), we obtain the following equation.

$$\begin{aligned}\log p(q|d) =& \sum_{i:c(q_i;d)>0} \log\left(1+\frac{c(q_i;d)}{\mu_1\cdot p(q_i|C)}\right)\\ &+\sum_{i:c(q_{j(i)};d)>0} \log\left(\frac{\sum_k c(q_k;d)+\mu_1}{\sum_k c(q_k,q_{j(i)};d)+\mu_2}\cdot\frac{\mu_2\cdot p(q_i|q_{j(i)},C)}{\mu_1\cdot p(q_i|C)}\right)\\ &+\sum_{i:c(q_i,q_{j(i)};d)>0} \log\left(1+\frac{c(q_i,q_{j(i)};d)}{\mu_2\cdot p(q_i|q_{j(i)},C)}\right)\\ &-\sum_{i:c(q_i,q_{j(i)};d)>0} \log\left(1+\frac{c(q_i;d)}{\mu_1\cdot p(q_i|C)}\right)\\ &+\sum_i \log\left(\frac{\mu_1\cdot p(q_i|C)}{\sum_k c(q_k;d)+\mu_1}\right).\end{aligned} \tag{12}$$

From (12), we define $MS_{DSLM\text{-}D}(q, d)$, a query-document scoring function adapted from the dependency structure language model using the Dirichlet smoothing method, as follows:

$$
\begin{aligned}
&MS_{DSLM\_D}(q,d) \\
&= \sum_{i:c(q_i;d)>0} \log\left(1+\frac{c(q_i;d)}{\mu_1 \cdot p(q_i|C)}\right) \\
&+ \sum_{i:c(q_{j(i)};d)>0} k\cdot\log\left(\frac{\sum_k c(q_k;d)+\mu_1}{\sum_k c(q_k,q_{j(i)};d)+\mu_2}\cdot\frac{\mu_2\cdot p(q_i|q_{j(i)},C)}{\mu_1\cdot p(q_i|C)}\right) \\
&+ \sum_{i:c(q_i,q_{j(i)};d)>0} k\cdot\log\left(1+\frac{c(q_i,q_{j(i)};d)}{\mu_2\cdot p(q_i|q_{j(i)},C)}\right) \\
&- \sum_{i:c(q_i,q_{j(i)};d)>0} k\cdot\log\left(1+\frac{c(q_i;d)}{\mu_1\cdot p(q_i|C)}\right) \\
&+ \sum_i \log\left(\frac{\mu_1\cdot p(q_i|C)}{\sum_k c(q_k;d)+\mu_1}\right),
\end{aligned}
\tag{13}
$$

where k is a constant parameter to control the influence of each part. Thus if $k = 0$ gives a unigram language model and $k = 1$ gives a fully dependency structured language model, different values of k give a mix of the two. In the formula, $p(q_i|q_{j(i)}, C)$ can also be zero because of the data sparseness problem, so we again apply the same smoothing to $p(q_i|q_{j(i)}, C)$ as follows:

$$
p(q_i|q_{j(i)},C) = \frac{\sum_d c(q_i,q_{j(i)};d)+\mu_3\cdot p(q_i|C)}{\sum_d\sum_k c(q_k,q_{j(i)};d)+\mu_3}.
$$

3. Interpolating Dependency Structure Language Model with Bigram Language Model

In the speech recognition field, many researchers have developed various grammar-based language models [5]-[7]. They have showed that grammar-based language models outperform the trigram language model. They also interpolated a grammar-based model with trigram model, and the interpolated model outperformed the original grammar-based model and trigram model. Chelba & Jelinek [6] and Roark [7] used the word-level interpolation. Charniak interpolated the probabilities of entire sentences [5]. This is a much less powerful technique than the word-level interpolation, but he still observed a significant gain in performance.

Considering these previous researches, we interpolate our dependency structure language model with a bigram language model. We interpolated the probabilities of the entire sentence (query q) as Charniak did. We also use a dependency structure language model, which is restricted to only bigram dependency (that is, assuming the dependency parse tree is linear) as a bigram language model. We define the interpolated model using the Jelinek-Mercer smoothing method as follows:

$$
\begin{aligned}
&MS_{DSLM-J-Interp}(q,d) \\
&= \alpha\cdot MS_{DSLM-J-Minipar}(q,d)+(1-\alpha)\cdot MS_{DSLM-J-Bigram}(q,d),
\end{aligned}
\tag{14}
$$

where α is a tunable constant parameter, $MS_{DSLM\text{-}J\text{-}Minipar}$ is a query-document scoring function in (9) using a dependency parse tree generated by Minipar, and $MS_{DSLM\text{-}J\text{-}Bigram}$ is a query-document scoring function in (9) that is restricted to only bigram dependency.

We also define the interpolated model using the Dirichlet smoothing method in the same manner.

IV. Experiment

1. Experiment Design

The goal of our experiment is to answer the following three questions:

Will the dependency structure language model be effective for information retrieval? To answer this question, we will compare the performance of the dependency structure language model with that of the state-of-the-art information retrieval methods, including the Okapi BM25 and recently proposed language models for information retrieval.

Will the dependency syntactic structure be more effective than a bigram model, which only models the co-occurrence of terms in language modeling for information retrieval? To answer this question, we will compare the results for the dependency structure language model using a dependency parse tree with the results for the same model using only bigram dependency (that is, assuming the dependency parse tree is linear) and the BBN method that models bigram production [1].

Will the interpolating dependency structure language model with bigram language model be effective? To answer this question, we will compare the result of the interpolation model (that is, DSLM-D-Interpolation and DSLM-JM-Interpolation) with the results of the other models.

We used two different TREC testing collections for evaluation: AP88 (Associated Press, 1988) and WSJ90-92 (Wall Street Journal from 1990 to 1992). The queries are TREC topics 202-250 (title field only) on TREC disks 2 and 3. We excluded the TREC topic 201 from the experiments because the topic's relevant documents are not included in the AP88 test collection. We used relatively small test collections (AP88 and WSJ90-92) because of the cost for parsing the entire collection. The dependency parse tree of the user query is obtained by the dependency parser (Minipar) at the search time, and the dependency relation information between the two terms in a document is obtained by the dependency parser at the indexing time.

In our experiment, k, λ_1 (or μ_1), λ_2 (or μ_2), λ_3 (or μ_3), and α are determined by performing a parameter tuning. This involves running the model with several different values and measuring the performance of the model. The best performing value is then chosen.

2. Baseline Methods

The three baseline methods are the Okapi BM25 [15], Zhai's language model [4], and the BBN method [1]. The formula for the Okapi BM25 (simplified version: k_2=0, k_3=∞, c=1) is given by

$$MS_{BM25}(q,d)=\sum_i \left(qtf_i \cdot \frac{tf_i}{k_1(1-b+b\frac{dl_d}{avdl})+tf_i} \cdot \log \frac{N-n_i+0.5}{n_i+0.5} \right),$$

where tf_i is the term frequency of q_i in document d, n_i is the document frequency for q_i, N is the number of documents in the collection, dl_d is the document length, $avdl$ is the average document length for all the documents in the collection, qtf_i is frequency of occurrence of the query term q_i within a specific query, and k_1 and b are determined by performing a parameter tuning.

In Zhai and Lafferty [4], three smoothing methods are compared. We use two smoothing methods (Jelinek-Mercer and Dirichlet) as our baseline methods. The formula, which uses the Jelinek-Mercer smoothing method, is given by

$$MS_{LM-J}(q,d)=\sum_{i:c(q_i;d)>0} \log\left(1+\frac{(1-\lambda)\cdot p_{ml}(q_i|d)}{\lambda\cdot p(q_i|c)}\right),$$

where λ is determined by performing a parameter tuning.

The formula applied Dirichlet smoothing method is given by

$$MS_{LM-D}(q,d)=\sum_{i:c(q_i;d)>0} \log\left(1+\frac{c(q_i;d)}{\mu\cdot p(q_i|C)}\right)+n\log\frac{\mu}{|d|+\mu},$$

where n is the number of query terms, $|d|$ is the document length, and μ is determined by performing a parameter tuning.

The BBN group suggested a hidden Markov model (HMM) that models bigram production [1]. The formula for the BBN method is given by

$$MS_{BBN-Brgram}(q,d) = \prod_i (a_0\cdot p(q_i|C)+a_1\cdot p_{ml}(q_i|d)+a_2\cdot p_{ml}(q_i|q_{i-1},d)),$$

where q_i is the current word of the query, $q_{i\text{-}1}$ is the previous word, and a_0, a_1 and a_2 are determined by performing a parameter tuning.

3. Experiment Results

The results on the AP88 test collection are shown in Tables 1 and 2. Table 3 shows the results of the WSJ90-92 test collection. In each table, we include the precision at different recall points and the average precision (non-interpolated).

In Table 1, *BM25* stands for Okapi BM25 formula, LM-D stands for Zhai's model applied to the Dirichlet smoothing method, DSLM-DB stands for a dependency structure language model that is restricted to only bigram dependency, DSLM-DM stands for a dependency structure language model using dependency parse tree generated by Minipar, and DSLM-DI stands for the interpolation model of DSLM-DM and DSLM-DB.

As seen from Table 1, the dependency structure language model applied to the Dirichlet smoothing method using the dependency parse tree generated by Minipar (DSLM-DM) has a significant gain in performance. DSLM-DM achieved about 5.99% improvement compared to BM25 in the average precision (the difference is not statistically significant). DSLM-DM also achieved about 7.67% improvement compared to Zhai's model (LM-D) in the average precision (the difference is statistically significant at the 0.05 level). And DSLM-DM achieved about 1.74% improvement compared to DSLM-DB (again, the difference is not statistically significant).

One notable fact is that the DSLM-DM performance includes the error of Minipar parsing in the given corpus. We hope we can achieve much better performance if we can improve the dependency parsing accuracy, which is about 85% in the current

Table 1. Results for AP88 collection for Dirichlet smoothing. (BM25: k_1=0.35, b=0.5; LM-D: μ= 8000; DSLM-DB: k=0.2, μ_1=8000, μ_2=50, μ_3=100000; DSLM-DM: k=0.35, μ_1=10000, μ_2=10, μ_3= 50000; DSLM-DI: α = 0.6)

Rec	BM25	LM-D	DSLM-DB	DSLM-DM	DSLM-DI
0.0	0.6631	0.6015	0.6163	0.6530	0.6550
0.1	0.5080	0.4949	0.5001	0.5375	0.5366
0.2	0.4244	0.4346	0.4386	0.4542	0.4598
0.3	0.3720	0.3848	0.4113	0.4001	0.4101
0.4	0.3312	0.3407	0.3679	0.3582	0.3653
0.5	0.2949	0.3153	0.3210	0.3133	0.3181
0.6	0.2269	0.2274	0.2536	0.2551	0.2586
0.7	0.1679	0.1737	0.1977	0.2034	0.2099
0.8	0.1194	0.1050	0.1201	0.1221	0.1244
0.9	0.0752	0.0630	0.0727	0.0733	0.0734
1.0	0.0520	0.0462	0.0496	0.0557	0.0556
AvgP	0.2754	0.2711	0.2869	0.2919	0.2959

Minipar system. In Table 1, DSLM-DI also has the best average precision. DSLM-DI achieved about 1.37% improvement compared to DSLM-DM. Thus, the table also shows that the interpolating dependency structure language model with bigram language model is as effective in information retrieval as it is in the speech recognition field.

In Table 2, LM-J stands for Zhai's model with the Jelinek-Mercer smoothing method, and BBN-B stands for the BBN method that models bigram production. Table 2 also shows that the dependency structure is more effective than the bigram in language modeling for information retrieval.

DSLM-JM achieved about a 9.35% improvement for Zhai's model with the Jelinek-Mercer smoothing method (LM-J) in average precision (statistically significant at the 0.05 level). DSLM-JM also achieved about 2.42% improvement over BBN-B (the difference is statistically significant at the 0.10 level). DSLM-JM achieved about 1.08% improvement for DSLM-JB (the difference is not statistically significant). DSLM-JI (the best parameter value is α = 0.7) has the best average precision.

Tables 3 and 4 show the results for the WSJ90-92 test collection. As seen from Table 3, DSLM-DM and DSLM-JM have significant gains in performance compared to LM-D and LM-J, respectively (statistically significant at the 0.10 level). DSLM-DI (α= 0.6) and DSLM-JI (α= 0.4) also have the best average precisions, respectively. So, in general, we can verify that the dependency structure language model is more effective than the conventional language modeling for information retrieval, and the interpolating dependency structure language model with bigram language model is as effective in information retrieval as it is in the speech recognition field.

Table 2. Results for AP88 collection for Jelinek-Mercer smoothing. (LM-J: λ=0.6; BBN-B: a_0=0.32, a_1=0.03, a_2=0.65; DSLM-JB: k=0.5, λ_1=0.8, λ_2=0.85, λ_3=0.999; DSLM-JM: k=0.4, λ_1=0.7, λ_2=0.05, λ_3=0.85; DSLM-JI: α=0.7)

Rec	LM-J	BBN-B	DSLM-JB	DSLM-JM	DSLM-JI
0.0	0.6058	0.5937	0.5938	0.6459	0.6426
0.1	0.4761	0.4919	0.5068	0.5026	0.5145
0.2	0.4109	0.4437	0.4481	0.4431	0.4575
0.3	0.3444	0.3590	0.3490	0.3563	0.3643
0.4	0.2786	0.3122	0.3191	0.3224	0.3326
0.5	0.2518	0.2788	0.2819	0.2963	0.2885
0.6	0.1953	0.2068	0.2141	0.2217	0.2254
0.7	0.1620	0.1764	0.1776	0.1841	0.1863
0.8	0.1205	0.1316	0.1366	0.1292	0.1354
0.9	0.0738	0.0811	0.0868	0.0787	0.0856
1.0	0.0482	0.0540	0.0586	0.0577	0.0622
AvgP	0.2480	0.2648	0.2683	0.2712	0.2782

Table 3. Results for WSJ90-92 collection for Dirichlet smoothing. (LM-D: μ=10000; DSLM-DM: k=0.15, μ_1=50000, μ_2=10, μ_3=100000; DSLM-DI: α =0.6)

Rec	LM-D	DSLM-DM	% change over LM-D	DSLM-DI
0.0	0.6505	0.6492	-0.20	0.6528
0.1	0.4601	0.4724	+2.60	0.4755
0.2	0.3572	0.3765	+5.13	0.3780
0.3	0.2575	0.2802	+8.10	0.2816
0.4	0.1764	0.2148	+17.88	0.2322
0.5	0.1482	0.1850	+19.90	0.2011
0.6	0.1195	0.1533	+22.05	0.1666
0.7	0.0674	0.0999	+32.53	0.1078
0.8	0.0559	0.0762	+26.64	0.0809
0.9	0.0360	0.0544	+33.82	0.0622
1.0	0.0226	0.0377	+40.05	0.0405
AvgP	0.1890	0.2101	+10.04	0.2172

Table 4. Results for WSJ90-92 collection for Jelinek-Mercer smoothing. (LM-J: λ=0.8; DSLM-JM: k=0.35, λ_1=0.7, λ_2=0.05, λ_3=0.95; DSLM-JI: α=0.4).

Rec	LM-J	DSLM-JM	% change over LM-J	DSLM-JI
0.0	0.5461	0.5619	+2.81	0.5924
0.1	0.4391	0.4566	+3.83	0.4676
0.2	0.3717	0.3782	+1.72	0.3897
0.3	0.2826	0.2991	+5.52	0.3196
0.4	0.2258	0.2529	+10.72	0.2517
0.5	0.1782	0.2103	+15.26	0.2211
0.6	0.1360	0.1586	+14.25	0.1597
0.7	0.0736	0.0783	+6.00	0.1016
0.8	0.0563	0.0548	-2.74	0.0723
0.9	0.0310	0.0287	-8.01	0.0430
1.0	0.0161	0.0169	+4.73	0.0283
AvgP	0.1951	0.2094	+6.83	0.2215

V. Conclusion

In the language model for information retrieval, unigram and bigram language models have some limitations to capture the underlying semantics in a document due to their inability to handle long-distance dependencies. In this paper, we propose a dependency structure language model to compensate for the weakness of the unigram and bigram language models in information retrieval. The dependency structure language model is based on the first-order dependency model and the dependency parse tree generated by a dependency parser. Thus, long-distance dependencies can be naturally handled by the dependency structure language model. We carried out some experiments to verify the proposed model. The experiments were performed on both AP88 and WSJ90-92 test collections. Based on the experiment results, we can draw the following conclusions:

1) Based on the comparison between the dependency structure language model and traditional language model and the Okapi BM25 method, we can conclude that the dependency structure language model for information retrieval is an effective retrieval method. In our experiments, the dependency structure model gives a better performance than both the traditional language model (statistically significant at the 0.05 level) and the Okapi BM25 method (the p-value is 0.186).

2) Based on the comparison between the dependency structure and bigram dependency in language modeling for information retrieval, we can also conclude that the dependency structure is more effective than the bigram in language modeling for information retrieval (not statistically significant).

3) Based on the comparison between the interpolation model and the dependency structure language model, we can also conclude that the interpolating dependency structure language model with bigram language model is also effective in information retrieval.

The disadvantage in using the dependency parser is that the computational cost of a dependency parse tree becomes high because the dependency parse tree of the user query is obtained by a dependency parser at the search time, and the co-occurrence information between the two terms are obtained by a dependency parser at the indexing time (in general, the cost of any language parsing is $O(n^3)$). To reduce this computational cost, we need to find a way that we do not rely on a full dependency parser, but use a more simplistic phrase chunker or partial parser in the future.

References

[1] D. Miller, T. Leek, and R. M. Schwartz, "A Hidden Markov Model Information Retrieval System," *Proc. 22nd Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1999, pp. 214–222.

[2] J. M. Ponte and W. B. Croft, "A Language Modeling Approach to Information Retrieval," *Proc. 21st Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval,* 1998, pp. 275–281.

[3] F. Song and W. B. Croft, "A General Language Model for Information Retrieval (Poster Abstract)," *Proc. of the 22nd Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval,* 1999, pp. 279–280.

[4] C. Zhai and J. Lafferty, "A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval," *Research and Development in Information Retrieval,* 2001, pp. 334–342.

[5] E. Charniak, "Immediate-Head Parsing for Language Models," *Proc. Thirty-Ninth Annual Meeting of the Association for Computational Linguistics and Seventeenth Int'l Conf. on Computational Linguistics,* 2001, pp. 116–123.

[6] C. Chelba and F. Jelinek, "Exploiting Syntactic Structure for Language Modeling," *Proc. Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth Int'l Conf. Computational Linguistics,* San Francisco, California, 1998, pp. 225–231.

[7] B. Roark, "Probabilistic Top-Down Parsing and Language Modeling," *Computational Linguistics,* vol. 27, no. 2, June 2001, 249–276.

[8] C. V. Rijsbergen, *Information retrieval*, Butterworths, 1979.

[9] J. Gao, J.-Y. Nie, G. Wu, and G. Cao, "Dependence Language Model for Information Retrieval," *Proc. 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004, pp. 170–177.

[10] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, A Wiley-Interscience Publication, 1973.

[11] C. Chow and C. Liu, "Approximating Discrete Probability Distributions with Dependence Trees," *IEEE Transactions on Information Theory,* vol. IT-14, May 1968, 462–467.

[12] D. Hays, "Dependency Theory: Formalism and Some Observations," *Language,* vol. 40, no. 4, 1964, 511–525.

[13] D. Lin, "Principa - An Efficient, Broad-Coverage, Principle Based Parser," *Proc. Fifteenth International Conference on Computational Linguistics,* COLING-ACL, 1994, pp. 109–126.

[14] MINIPAR: 1998. http://www.cs.ualberta.ca/_lindek/minipar.htm.

[15] S. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3," *Proc. Second Text Retrieval Conf. (TREC-3)*, 1995.

Changki Lee has received the BS degree in computer science from KAIST, Korea in 1999. He received the MS and PhD degrees in computer engineering from POSTECH, Korea in 2001 and 2004. Since 2004, he has been with Electronics and Telecommunications Research Institute (ETRI), Korea, as a Senior Member of Research Staff. He has served as a reviewer for some international journals such as Information System and Information Processing & Management. His research interests are in natural language processing, information retrieval, information extraction, question answering, spoken language understanding, and semantic web.

Gary Geunbae Lee received the BS and MS degrees in computer engineering from Seoul National University in 1984 and 1986. He received the PhD in computer science from UCLA in 1991, where he was a research scientist. He was an Assistant Professor from 1992 to 1996, and was an Associate Professor beginning in 1997 at POSTECH, where in 2004, he was promoted to a full Professor. He has authored more than 100 papers in international journals and conferences and has served as a technical committee member and reviewer for several international conferences such as ACL, COLING, ACM SIGIR, IRAL, EMNLP, and others. His current research interests include natural language processing, biological text mining, spoken language understanding, and TTS systems.

Myung-Gil Jang received the BS and MS degrees in computer science & statistics from Pusan National University, Korea in 1988 and 1990. He received the PhD degree in information science from Chungnam National University in 2002. He was with System Engineering Research Institute (SERI), Korea, from 1990 to 1997 as a researcher. Since 1998, he has been with ETRI, Korea, as a Senior/Principle Member of Research Staff. His research interests are natural language processing, information retrieval, question answering, knowledge & dialogue processing, media retrieval/management, and semantic web.