Recursive Markov Chain as a Stochastic Grammar

Young S. Han, Young C. Park, Key-Sun Choi
Department of Computer Science
Korea Advanced Institute of Science and Technology
Taejon, 305-701, Korea
yshan@csking.kaist.ac.kr

Abstract

A grammar network that resembles the RTN, but takes the topology of Markov Model is introduced. This new grammar model captures the linguistic patterns from the stochastic process of the patterns. The linguistic patterns may be derived with minimal human involvement. When grammar is viewed in Markov process, we are given a rich set of tools that can be applied to the analysis and generation of languages.

In this paper an informal introduction to the model is made and evaluation problem is discussed in detail. An experiment shows that the Markov nets with stack can properly model natural languages.

1 Introduction

The position of probabilistic approach to natural language processing is no longer arguable in the field. The main endeavour is to discover and make use of the well structured probabilistic behavior of languages. A probabilistic parser is particularly favored for its robustness, adaptability, and domain dependency among others. If a grammar cannot capture the most general account of a language, it may well be faithful to a regional account of a language for a practical effectiveness. In this regard none other than empirical parsers is best appreciated thus far.

Probabilistic methods have been extensively studied in the area of automatic speech processing. They compute the probability that a successive application of rewrite rules returns a word string(Jelinek 91). In general there are at least three problems to be considered in designing and working with a Stochastic Grammar.

1. Definition of probability with respect to a linguistic structure must be made first. The nature of probability must be clearly defined along with the level of linguistic objects.
2. Evaluation of the parse trees must be both consistent and intuitive. The purpose of using probability is eventually to evaluate total or partial results from the parse. It must be applicable locally as well as globally, and serve as an evidence for the decision making.
3. Acquisition of probability should support the model with high precision and be as much cost effective as possible.

In this paper we introduce a probabilistic parser that addresses different directions in the above issues. One of the popular methods exemplified by (Lari 90) does not take still valuable information into account, namely the transitional probability between two adjacent linguistic items. Then we will look at how a grammar of transition network type can deal with evaluation problem. Through hierarchical networks a grammar is seen as a set of rules specifying different levels of abstraction over the sentences. When the sequence(occurrence) of linguistic tokens is seen as a Markov Process, the network becomes Markov chain in which first order dependency between context free linguistic objects can be captured. A reasonable evaluation of parse tree can be made even when the grammar is not in Chomsky Normal Form. The argument of this type is hard to make sound, nonetheless we present an experimental evidence to support the claim.

Acquisition of probabilities, estimation of the grammar from Corpus, and parse control strategy are all practical issues. Regarding the mechanism of parsing under the model there can be several options including one that uses a heuristic function based on the probability of known (already parsed) portion of a path. In Markov models, the search problem is well defined and is no longer a problem as it can be solved in polynomial time(Rabiner 89).In our case, however, the size of network can grow so big that even the polynomial complexity may be a severe burden. The heuristic measure named **RPS** (Return Path Probabilistic Search) makes use of the unique property of recurrent Markov chain(to be published). It helps parse decisions be made for a better path with higher probability using only limited information.

2 Evaluation of Parse Trees

While the need of evaluating trees is diverse, our interest is in selecting the most probable parse tree of poten-

tially numerous alternatives. By tree's being probable, we mean the tree is in fact correct not only syntactically but also semantically. Assuming no more knowledge than syntactic rules available in the system, we do our best by establishing quantitative correlation between syntactic structures and the occurrences of correct parses. Thus, given a sentence, W, we want to find

$$\max[P(\mathcal{S}_i|W)], \quad \mathcal{S}_i \in \Gamma ,$$

where Γ is a set of syntactically correct trees.

The probability that a sentence, W, may be generated by a grammar, $\mathcal{G}$, is

$$P(\mathcal{G} \vdash W) = \sum_i^n P(\mathcal{S}_i|W) .$$

More generally if we let $P(H < i, i+n >)$ denote $P(H \rightarrow w_i w_{i+1} \cdots w_{i+n})$, the probability that starting with the non terminal, H, the successive application of grammar rules has produced the string, $w_i w_{i+1} \cdots w_{i+n}$, a well know Inside Algorithm(Baker 79) that evaluates trees gives the following equation,

$$P(H < i, i+n >) = \sum_{N_1 N_2} P(H \rightarrow N_1 N_2) \times$$
$$[\sum_{j=1}^{n} P(N_1 < i, i+j-1 >) \times$$
$$P(N_2 < i+j, i+n >)] .$$

Notice Chomsky Normal Form (CNF, hereafter) is assumed in the above formula. It is the sum of all the combinations of sub trees. One popular algorithm for searching for the most probable tree is Viterbi algorithm. But when the grammar is not built upon CNF having rules of more than two non terminals with associated probability, the grammar will fail to strictly reflect at least two of many aspects with regard to evaluating parse trees.

- Probable parse trees are more than successive application of rewrite rules.
- Probable parse trees are not necessarily sensitive to the number of nodes of each tree.

There are two choices to deal with the above. The one is to have an efficient way of converting it to CNF including the proper adjustment of probabilities, when an application prefers non CNF. The other is to come up with a way to avoid the problems in grammars of non CNF. It is the latter direction that this paper brings up along with other reasons to discuss later.

The first point of the list is obvious because a parse tree is a set of horizontally and vertically ordered linguistic objects. What is caught in Inside Algorithm is vertical dependency, but the other incompletely. In CNF, horizontal dependency is reasonably maintained as every rule is ground up to two elements. There, however, exists one dependency between the two elements of each rule that is left uncovered. Now we ask if it is safe to desert the dependency at all. It will be safe if for a pair of H rules the following holds,

$$\frac{P(H \rightarrow N_1 N_2)}{P(H \rightarrow N_3 N_4)} = \frac{P(N_2|N_1)}{P(N_4|N_3)},$$

where $P(N_2|N_1)$ denotes the dependency of N_2 to N_1.

Obviously this is too strong to stand up at all times. We redefine probable trees to be highly probable in both vertically and horizontally in a tree such that the probability that a substring, $w_i w_{i+1} \cdots w_{i+n}$, may be generated by a stochastic grammar of CNF is computed as follows:

$$\Pr(H < i, i+n >) = \sum_{N_1 N_2} P(H \rightarrow N_1 N_2) \times$$
$$[\sum_{j=1}^{n} \Pr(N_1 < i, i+j-1 >) \times$$
$$\Pr(N_2 < i+j, i+n >)] P(N_2|N_1) .$$

More generally, for non CNF,

$$\Pr(H < i, i+n >) =$$
$$\sum_{N_1 N_2 \cdots N_p N_q} P(H \rightarrow N_1 N_2 \cdots N_p N_q) \times$$
$$[\sum_{j=1}^{n} \Pr(N_1 < i, i+j-1 >) \times$$
$$\sum_{k=1}^{n-1} \Pr(N_2 < i+j, i+j+k-1 >) \cdots$$
$$\sum_{m=1}^{n-q+1} \Pr(N_p < i+j+k+\cdots+l,$$
$$i+j+k+\cdots+l+m-1 >) \times$$
$$\Pr(N_q < i+j+k+\cdots+m, i+n >)] \times$$
$$\prod_{t=1}^{q-1} P(N_{t+1}|N_t) .$$

As the following holds,

$$\sum_i P(N_i|N_j) = 1 ,$$

the well definedness of the SCFG the notion of which is due to (Jelinek 91) is not affected. That is,

$$\sum_{n=1}^{\infty} \sum_{w_1 w_2 \cdots w_n} P(\Psi \vdash w_1 w_2 \cdots w_n) = 1 , \; w_i \in V ,$$

where V is the set of terminal symbols.

The model to be introduced in the next section observes the above computation of probability of trees.

In a grammar of CNF, a sentence of n tokens will have the trees whose maximum depth is $n-1$, and the minimum is $\log_2 n$. All the trees are fairly evaluated since they have the same number of nodes which is $2n-1$.

To see how an evaluation in a non CNF grammar can go wrong, consider the example in figure 1, it is less unlikely for b to be more probable than a simply because a has more nodes to compute. Depending on the domain of discourse, the number of nodes in a tree may command positive relation with the probability of its being correct. With no sound result from psycholinguistics, we have to have evaluation normalized in the number of nodes.

In the following section a network grammar model that is flexible, robust, and intuitive is introduced followed by an experimental result.

3 Recursive Markov Chains as a Stochastic Grammar

Linguistic patterns occur repeatedly with frequency of different expectations. such a pattern may occur with a certain rate, but the time of recurrence is not definitely predictable. The time between any moment after a pattern occured and the next occurrence of the pattern has the same distribution as the time interval between the two occurrences. Thus, the distribution between occurrences is memoryless, which, in our case, can be seen as geometric distribution if we consider discrete time scale. In fact, this is true for any pair of linguistic patterns as well as recurring patterns.

For each linguistic pattern of any nature a separate stochastic process is assumed. These stochastic processes are dependent of some other patterns, thus while it is possible to view a language consisting of a set of independent stochastic processes, more constructive approach is to set up the processes within dependency.

There are patterns corresponding to sentential structure, patterns for clausal structures, and so on. These patterns provide different level of abstraction over the sentences. When they are put in proper dependency, we get Recursive Markov Model. Recursive Markov Model consists of a Markov model and stacks. It takes much the same configuration as ordinary RTN except that the network observes the constraints of recurrent Markov model.

The transition in the network grammar represents the dependency between two linguistic items. In other words, the grammar asserts the independencies among other linguistic items, which can be very useful in decision theoretic terms(Heckerman 91).

A similar but different work can be found from TINA project in MIT(Seneff 89) which employs a probabilistic grammar of network type.

The whole business of employing Markov model is to gain a new insight into the problem as well as to use

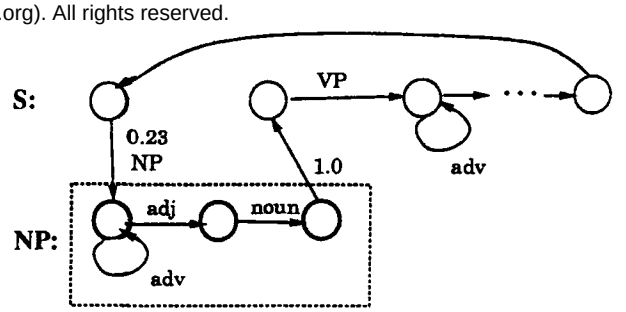

Figure 2: Combining subnets together

rich techniques already available in the more evolved network concept. The reasons that we resort to networks instead of rules, include the brittleness, exhaustiveness, and the difficult expandability of rules. A Recursive Markov Grammar $\mathcal{G}$ is specified by V, N Γ, and E. where V is a set of terminal symbols, N is a set of nonterminals, Γ is a set of states, and E is a set of weighted edges. All the parameters together uniquely determine a language. Besides, E and N must conform the topology of recurrent Markov chain that is irreducible and recurrent non null. Notice there is no explicit push and pop arc in the network(see figure 6) though operational semantics has them implicitly. The philosophy is that each sub network is an independent unit of its own right covering as much independent linguistic process as possible in Markovian sense. Consequently each sub network sees the world of domain only through its states. Conceptually the final state is directly linked to the start state , then the corpus is not a set of sentences any more, but a series of events. Different networks cover just different levels of abstraction over the same corpus.

The generative power of the network grammar is equivalent to that of a CFG. The grammar *properly* induces all the sentences consisting only of terminal symbols in a language that the grammar specifies. It is also quite obvious the grammar well defines a language in that

$$\sum_{n=1}^{\infty} \sum_{w_1 w_2 \cdots w_n} P(\mathcal{G} \vdash w_1 w_2 \cdots w_n) \;=\; 1 \;,$$

since every decision point is at transitions from a state, total sum of which is 1,

$$\sum_i P(S_i|S_j) \;=\; 1, \;\; S_i, S_j \in N \;.$$

Here is a bit tricky. As stated earlier, the whole grammar, $\mathcal{G}$, consists of many seemingly independent sub networks. We, in fact, can stitch the subnets into a big one (see figure 2).

Notice the probabilities of terminal symbol occurrences may be maintained separately from the main network, for the purpose of simplicity.

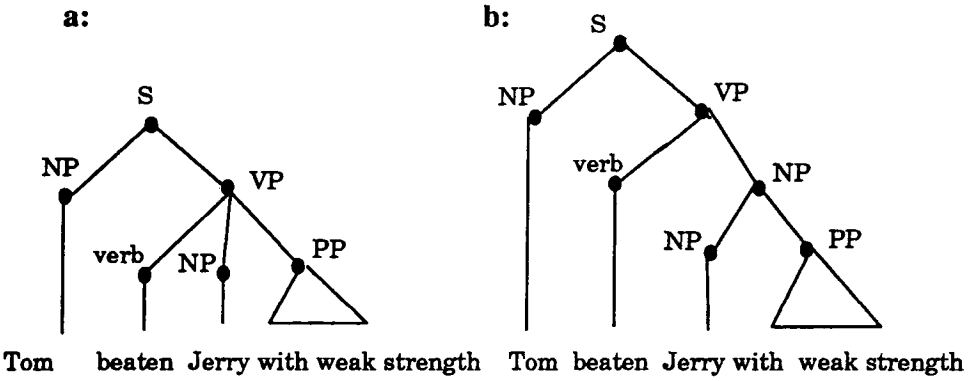

Figure 1: Evaluation Problem

Once the subnets are synthesized into one, the evaluation is just the product of transitional probabilities of a path that corresponds to a tree. A valid path forms a tree structure as in figure 3.

As figure 3 shows every nonterminal transition makes a branch (consider syntactic categories as terminals for simplicity). Given a sentence of size, n, there can be a tree of at most $n-1$ depth. As a tree has more nonterminals it will be less probable than one with fewer nonterminals. We can change our networks accordingly as in CNF, but here we introduce an evaluation function that works apart from the well definedness of the grammar, but normalize the effect of nonterminals. In CNF, every two elements form a nonterminal move, but in the proposed function, every nonterminal transition will be coalesced into terminals. After the evaluation is done, there will be only terminals and transitions between them, and the product of transitions will be the final outcome. A clear observation is

$$E(\mathcal{G} \vdash w_1 w_2 \cdots w_n) \neq P(\mathcal{G} \vdash w_1 w_2 \cdots w_n) \ ,$$

where
$E(\mathcal{G} \vdash w_1 w_2 \cdots w_n)$ is the total sum of evaluations over the paths in which $\mathcal{G}$ generates $w_1 w_2 \cdots w_n$.

As the likelihood of the occurrence of a tree is not necessarily in proportion to the likelihood of its being correct in our model, we do not mind the inequality above. Let $E(H < i, i+n >)$ denote the evaluation over the substring from i to $i+n$. A similar argument to Inside Algorithm can be drawn, and we state only the result.

$$\begin{aligned} E(H < i, i+n >) = \sum_{i,j,k} [P(H|s) \\ E(N_1 < i, i+j >) \times \\ E(N_2 < i+j+\phi, i+j+k+\phi >) \cdots \\ E(N_q < i+j+k+\phi, i+n >) \sum_{l=1}^{q} P(t_l|s)]^{\frac{n+1}{n+2}} \ , \end{aligned}$$

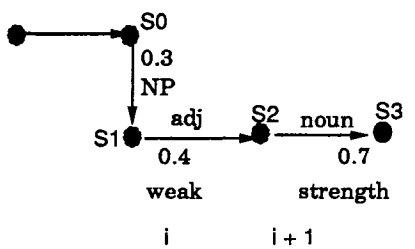

Figure 4: Coalescence of Non terminal

where
$H : \{N_1, N_2, \cdots, N_p, t_1, t_2, \cdots, t_q\}$,
$p + q = n, \ 0 \leq \phi \leq q$,
$s \in \Gamma, \ t \in V$.

H is a network that includes p nonterminals and q terminals. The above definition assumes a case starting and ending with non terminals for simplicity. Figure 4 illustrates a simple computation of the function.

$$\begin{aligned} E(NP < i, i+1 >) &= [P(NP|s_0)P(adj|s_1) \\ &\quad P(noun|s_2)] + \cdots \\ &= [0.3 \times 0.4 \times 0.7]^{\frac{2}{3}} + \cdots \end{aligned}$$

Now define,

$$E(H < i, i+n >) = \sum_j E(\lambda_j | H < i, i+n >) \ .$$

$E(\lambda_j | H < i, i+n >)$ is an evaluation of a particular partial path that starts with H and covers the substring. As far as parsing is concerned, we are interested in getting,

$$\max_j [E(\lambda_j | H < i, i+n >)] \ .$$

Now how valid the evaluation function E is can only be judged probabilistically just as in CNF.

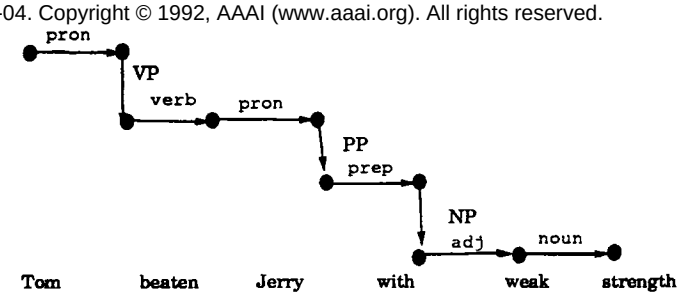

Figure 3: Path and Tree

To access the next state transitions from a state return paths of the next states are examined. Experimental results show that the accessment of the next states based on the return paths promises better probability for better paths. In particular, return paths are normalized in their size as in

4 Experiments

Our experiment is aimed to see how the proposed model captures the syntactic phenomena, and how the evaluation can work. The 300 sentences are selected such that they reflect the major structural types of American English provided by MATE-EK machine translation project. They are automatically parsed by a Tomita parser. Manual analysis over the trees gave the networks including S, NP, AP, ADP, PP, NCL(Noun clause), ADCL(Adverbial clause), and ACL(Relative clause). There can be numerous variations in constructing the networks depending on the applications.

To see if the evaluation function makes sense, see figure 5. Lots of simplification has been made to stress the difference between the two trees. It happened that in this experiment PP-attachment to NP was more probable than to verb. For tree, a, the outline computation is

$$(0.6415 \times 0.6)^{\frac{1}{2}} \times (X \times 0.1765)^{\frac{3}{4}} = E(a),$$

and for b is

$$(0.6415 \times 0.6 \times (X \times 0.1963)^{\frac{3}{4}})^{\frac{4}{5}} = E(b).$$

Obviously,

$$E(a) < E(b) .$$

It is easy to observe that the distribution of probabilities are so strong that the network is indeed a proper model of the linguistic activities (see figure 6).

5 Conclusion

In this paper we made a brief parsing model that is a part of our robust natural language processing project. It is now under implementation stage. Mostly the model was posed within the problems of evaluation. A grammar in CNF covers most dependencies, but the first order dependency between nonterminals of a rule is not covered. A grammar in non-CNF faces all the problems, but the coalescence of non terminals will help get away with the problem in grammars of non-CNF. This was a problem raised in developing Recursive Markov Grammar, which is another network grammar formalism based on empirical data. Future extension includes augmented edges with full utility of conditions and actions. For a practical use, the study on the automatic acquisition of grammar from corpus should be accompanied.

References

[1] Baker, J. K. 1979. Trainable grammars for speech recognition. Proceedings of Spring Conference of the Acoustical Society of America, 547-550. Boston, MA.

[2] Dougherty, Edward R. 1988. *Mathematical Methods for Artificial Intelligence and Autonomous Systems.* Prentice Hall.

[3] Heckerman, David. 1991. *Probabilistic Similarity Networks.* ACM Doctoral Dissertation Award 1990. MIT Press.

[4] Jelinek, F.; Lafferty John D. 1991. Computation of the Probability of Initial Substring Generation by Stochastic Context-Free Grammar. *Association for Computational Linguistics.*

[5] Lari, K.; Young S. J. 1990. The estimation of stochastic context free grammars using the Inside-Outside Algorithm. *Computer speech and Language*

[6] Molloy, Michael. 1989. *Fundamentals of Performance Modeling.* MacMillan Publishing Company.

[7] Rabiner, Lawrence R. 1989. A tutorial on Hidden Markov Models and selected Applications in Speech Recognition. Proceedings of the IEEE 77(2). Feb.

[8] Seneff, Stephanie. 1989. TINA: A Probabilistic Syntactic Parser for speech Understanding Systems. Proceedings of Speech and Natural Language Workshop February.

[9] Wetherell C. S. 1980. Probabilistic Languages: A review and some open questions, *Computing Surveys* 12(4).

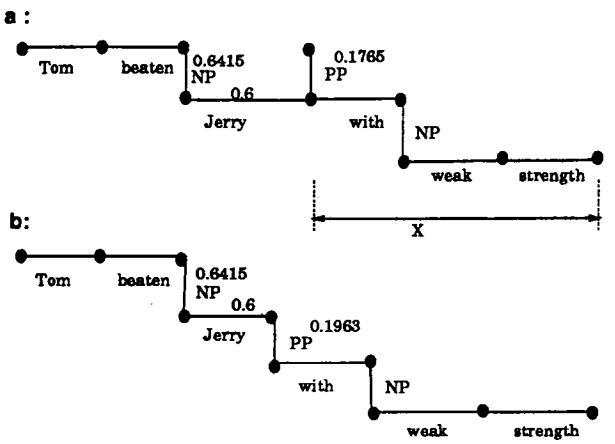

Figure 5: Evaluation Example

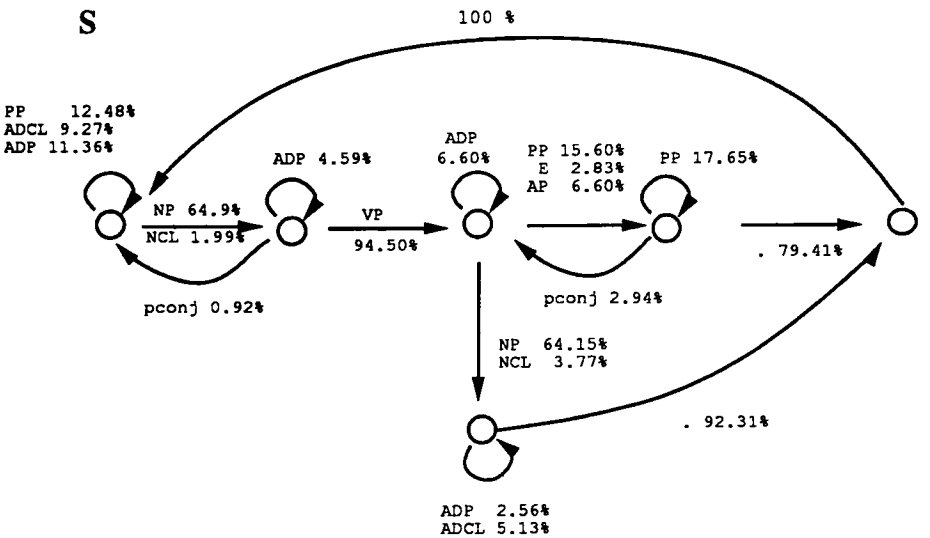

Figure 6: Sentence Model

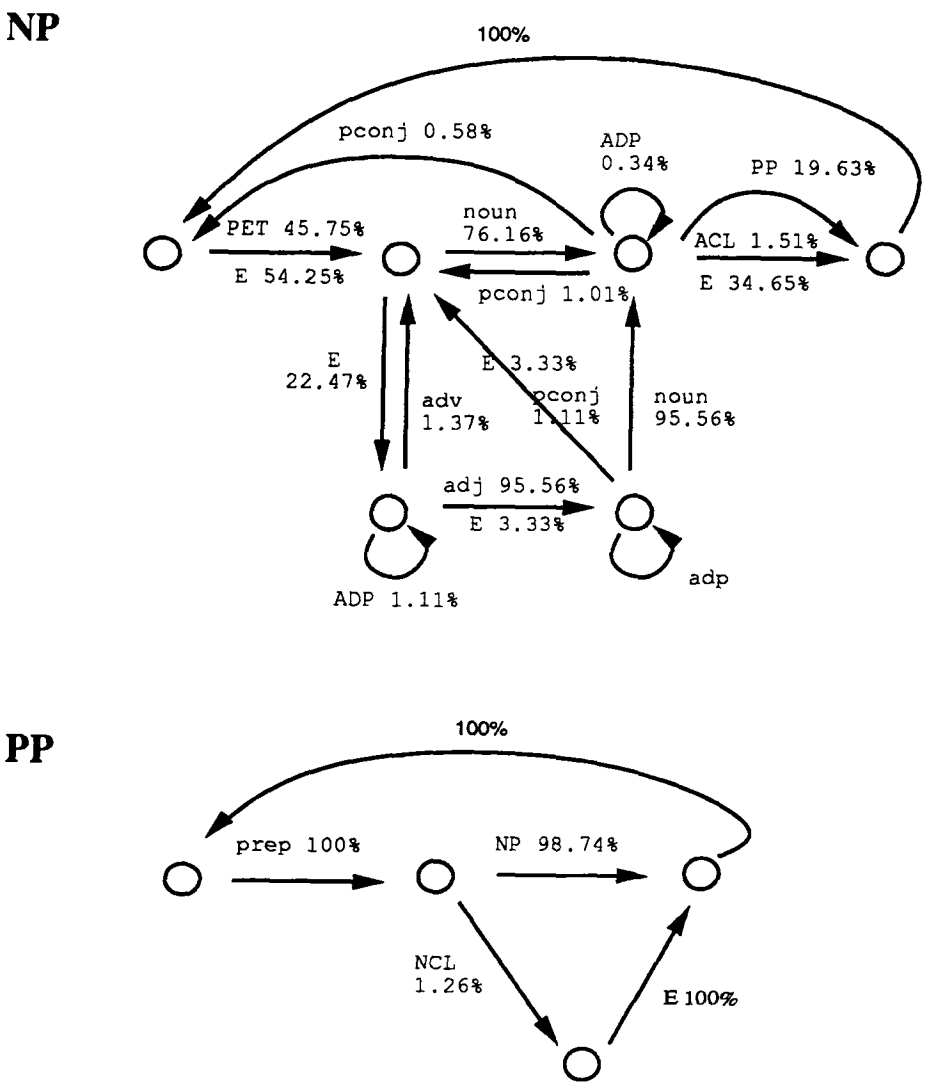

Figure 7: Noun Phrase and PP Model