PROBABILISTIC LR PARSING FOR SPEECH RECOGNITION

J.H.Wright and E.N.Wrigley

Engineering Mathematics, University of Bristol, U.K.

Abstract

An LR parser for probabilistic context-free grammars is described. Each of the standard versions of parser generator (SLR, canonical and LALR) may be applied. A graph-structured stack permits action conflicts and allows the parser to be used with uncertain input, typical of speech recognition applications. The sentence uncertainty is measured using entropy and is significantly lower for the grammar than for a first-order Markov model.

1. INTRODUCTION

1.1 *Background*

The automatic recognition of continuous speech requires more than signal processing and pattern matching: a model of the language is needed to give structure to the utterance. At sub-word level, hidden Markov models [1] have proved of great value in pattern matching. The focus of this paper is modelling at the linguistic level. Markov models are adaptable and can handle potentially any sequence of words [2]. Being probabilistic they fit naturally into the context of uncertainty created by pattern matching. However, they do not capture the larger-scale structure of language and they do not provide an interpretation. Grammar models capture more of the structure of language, but it can be difficult to recover from an early error in syntactic analysis and there is no watertight grammar.

A systematic treatment of uncertainty is needed in this context, for the following reasons:

(1) some words and grammar rules are used more often than others;

(2) pattern matching (whether by dynamic time warping, hidden Markov modelling or multi-layer perceptron [3]) returns a degree of fit for each word tested, rather than an absolute discrimination; a number of possible sentences therefore arise;

(3) at the end of an utterance it is desirable that each of these sentences receive an overall measure of support, given all the data so that the information is used efficiently.

The type of language model which is the focus of this paper is the probabilistic context-free grammar (PCFG). This is an obvious enhancement of an ordinary CFG, the probability information initially intended to capture (1) above, but as will be seen this opens the way to satisfying (2) and (3). An LR parser [4,5] is used with an adaptation [6] which enlarges the scope to include almost any practical CFG. This adaptation also allows the LR approach to be used with uncertain input [7], and this approach enables a grammar model to interface with the speech recognition front end

as naturally as does a Markov model.

1.2 *Probabilistic Context-Free Grammars*

A "probabilistic context-free grammar (PCFG)" [8-10] is a 4-tuple <N,T,R,S> where N is a nonterminal vocabulary including the start symbol S, T is a terminal vocabulary, and R is a set of production-rules each of which is a pair of form $\langle A \rightarrow \alpha, p \rangle$, with $A \in N$, $\alpha \in (N \cup T)^*$, and p a probability. The probabilities associated with all the rules having a particular nonterminal on the LHS must sum to one. A probability is associated with each derivation by multiplying the probabilities of those rules used, in keeping with the context-freeness of the grammar.

A very simple PCFG can be seen in figure 1: the symbols in uppercase are the nonterminals, those in lowercase are the terminals (actually preterminals) and λ denotes the null string.

2. LR PARSING FOR PROBABILISTIC CFGs

The LR parsing strategy can be applied to a PCFG if the rule-probabilities are driven down into the parsing action table by the parser generator. In addition, one of the objectives of using the parser in speech recognition is for providing a set of prior probabilities for possible next words at successive stages in the recognition of a sentence. The use of these prior probabilities will be described in section 3.1. In what follows it will be assumed that the grammars are non-left-recursive, although null rules are allowed.

2.1 *SLR Parser*

The first aspect of parser construction is the closure function. Suppose that I is an SLR kernel set consisting of LR(0) items of the form

$$\langle A \rightarrow \alpha \cdot \beta,\ p \rangle$$

The item probability p can be thought of as a posterior probability of the item given the terminal string up to that point. The computation of closure(I) requires that items

$$\langle B \rightarrow \cdot \gamma_r,\ p_B p_r \rangle$$

be added to the set for each rule $\langle B \rightarrow \gamma_r, p_r \rangle$ with B on the LHS, provided $p_B p_r$ exceeds some small probability threshold ϵ, where p_B is the total probability of items with B appearing after the dot (in the closed set).

New kernel sets are generated from a closed set of items by the goto function. If all the items with symbol $X \in (N \cup T)$ after the dot in a set I are

$$\langle A_k \rightarrow \alpha_k \cdot X \beta_k,\ p_k \rangle \quad \text{for } k=1,\ldots,n_X, \quad \text{with } p_X = \sum_{k=1}^{n_X} p_k$$

then the new kernel set corresponding to X is

$$\{\langle A_k \rightarrow \alpha_k X \cdot \beta_k,\ p_k/p_X \rangle \quad \text{for } k=1,\ldots,n_X\}$$

and goto(I,X) is the closure of this set. The set already exists if there

is another set which has the same number of elements, an exact counterpart for each dotted item, and a probability for each item that differs from that for its counterpart in the new set by at most ϵ.

Starting from an initial state I_0 consisting of the closure of

$$\{\langle S' \rightarrow \cdot S,\ 1\rangle\}$$

where S' is an auxiliary start symbol, this process continues until no further sets are created. They can then be listed as $I_0, I_1, \ldots$.

Each state set I_m generates state m and a row in the parsing tables "action" and "goto". The goto table simply contains the numbers of the destination states, as for the deterministic LR algorithm, but the action table also inherits probabilistic information from the grammar.

(1) For each terminal symbol b, if there are items in I_m such that the total $p_b > \epsilon$, and the shift state n is given by $goto(I_m, b) = I_n$, then

$$action[m,b] = \langle shift\text{-}to\text{-}n,\ p_b\rangle$$

(2) For each nonterminal symbol B, if $p_B > \epsilon$ and $goto(I_m, B) = I_n$ then

$$goto[m,B] = n$$

(3) If $\langle S' \rightarrow S\cdot,\ p\rangle \in I_m$ then $action[m,\$] = \langle accept,\ p\rangle$

(4) If $\langle B \rightarrow \gamma\cdot,\ p\rangle \in I_m$ where $B \neq S'$ then

$$action[m, FOLLOW(B)] = \langle reduce\text{-}by\ B \rightarrow \gamma,\ p\rangle$$

For the very simple grammar shown in figure 1 the parsing tables turn out as shown in figure 2, with shift-reduce optimisation [4,5] applied. The probability of each entry is underneath.

The range of terminal symbols which can follow a B-reduction is given by the set FOLLOW(B) which can be obtained from the grammar by a standard algorithm [4]. For a probabilistic grammar, the probability p attached to the reduce item cannot be distributed over those entries because when the tables are compiled it is not determined which of those terminals can actually occur next in that context, so the probability p is attached to the whole range of entries.

The probability associated with a shift action is the prior probability of that terminal occurring next at that point in the input string (assuming no conflicts). Completing the set of prior probabilities involves following up each reduce action using local copies of the stack until shift actions block all further progress. The reduce action probability must be distributed over the shift terminals which emerge. This is done by allocating this probability to the entries in the action table row for the state reached after the reduction, in proportion to the probability of each entry. Some of these entries may be further reduce actions in which case a similar procedure must be followed, and so on.

2.2 *Canonical LR Parser*

For the canonical LR parser each item possesses a lookahead distribution:

$$\langle A \rightarrow \alpha\cdot\beta,\ p,\ (P(a_i))_{i=1,\ldots,|T|}\rangle$$

The closure operation is more complex than for the SLR parser, because of the propagation of lookaheads through the non-kernel items. The items to be added to a kernel set to close it take the form

$$\langle B \rightarrow \cdot\gamma_r,\ p_B p_r,\ (P_B(a_j))_{j=1,\ldots,|T|} \rangle$$

so that all the items with B after the dot are then

$$\langle A_k \rightarrow \alpha_k \cdot B\beta_k,\ p_k,\ (P_k(a_i))_{i=1,\ldots,|T|} \rangle \quad \text{for } k=1,\ldots,n_B$$

and

$$P_B(a_j) = \sum_{k=1}^{n_B} \frac{p_k}{p_B} \sum_{i=1}^{|T|} P^F(\beta_k a_i, a_j) P_k(a_i)$$

where $P^F(\beta_k a_i, a_j)$ is the probability of a_j occurring first in a string derived from $\beta_k a_i$, which is easily evaluated. A justification of this will be published elsewhere. The lookahead distribution is copied to the new kernel set by the goto function.

The first three steps of parsing table construction are essentially the same as for the SLR parser. In step (4), the item in I_m takes the form

$$\langle B \rightarrow \gamma\cdot,\ p,\ (P(a_i))_{i=1,\ldots,|T|} \rangle \quad \text{where } B \neq S'$$

The total probability p has to be distributed over the possible next input symbols a_i, using the lookahead distribution:

$$\text{action}[m, a_i] = \langle \text{reduce-by } B \rightarrow \gamma,\ pP(a_i) \rangle$$

for all i such that $pP(a_i) > \epsilon$. The prior probabilities during parsing action can now be read directly from the action table.

2.3 *LALR Parser*

Merging the states of the canonical parser which differ only in lookaheads for each item causes the probability distribution of lookaheads to be lost, so for the LALR parser the LR(1) items take the form

$$\langle A \rightarrow \alpha\cdot\beta,\ p,\ L \rangle \quad \text{where } L \subseteq T.$$

The preferred method for generating the states as described in [4] can be adapted to the probabilistic case. Reduce entries in the parsing tables are then controlled by the lookahead sets, with the prior probabilities found as for the SLR parser.

2.4 *Conflicts and Interpretation*

An action conflict arises whenever the parser generator attempts to put two (or more) different entries into the same place in the action table, and there are two ways to deal with them. The first approach is to resolve each conflict [11]. This is a dubious practice in the probabilistic case because there is no clear basis for resolving the probabilities of the actions in conflict. The second approach is to split the stack and pursue all options, conceptually in parallel. Tomita [6] has devised an efficient enhancement of the LR parser which operates in this way. A graph-structured stack avoids duplication of effort and maintains (so far as

possible) the speed and compactness of the parser. With this approach the LR algorithm can handle almost any practical CFG, and is highly suited to probabilistic grammars, the main distinction being that a probability becomes attached to each branch.

The generation and action of the probabilistic LR parser can be supported by a Bayesian interpretation. This is in keeping with the further adaptation of the algorithm to deal with uncertain input.

3. UNCERTAIN INPUT DATA

3.1 *Prediction and Updating Algorithm*

The situation envisaged for applications of the probabilistic LR parser in speech recognition is depicted in figure 3. The parser forms part of a linguistic analyser whose purpose is to maintain and extend those partial sentences which are compatible with the input so far. With each partial sentence there is associated an overall probability and partial sentences with very low probability are suspended. It is assumed that the pattern matcher returns likelihoods of words, which is true if hidden Markov models are used. Other methods of pattern matching return measures which it is assumed can be interpreted as likelihoods, perhaps via a transformation.

let Γ_s^m (s=1,2,...) represent partial sentences up to stage m (the stage denoted by a superscript). let D^m represent the data at stage m, and $(D)^m$ represent all the data up to stage m. Each branch Γ_s^m predicts words a_j^m (perhaps via the LR parser) with probability $P(a_j^m|\Gamma_s^{m-1})$, so the total prior probability for each word a_j^m is

$$P(a_j^m|(D)^{m-1}) = \sum_s P(a_j^m|\Gamma_s^{m-1})P(\Gamma_s^{m-1}|(D)^{m-1})$$

Using Bayes' theorem the posterior probabilities of the words are

$$P(a_j^m|(D)^m) = \frac{P(D^m|a_j^m)P(a_j^m|(D)^{m-1})}{\sum_i P(D^m|a_i^m)P(a_i^m|(D)^{m-1})}$$

where $P(D^m|a_j^m)$ is the likelihood. If we define the extended branch Γ_{sj}^m as $\Gamma_s^{m-1}\&a_j^m$ then after some manipulation the probability of this is

$$P(\Gamma_{sj}^m|(D)^m) = \frac{P(a_j^m|\Gamma_s^{m-1})P(\Gamma_s^{m-1}|(D)^{m-1})}{P(a_j^m|(D)^{m-1})} P(a_j^m|(D)^m) \qquad (1)$$

This shows that the posterior probability of a_j^m is distributed over the extended partial sentences in proportion to their root sentences's contribution to the total prior probability of that word. If $P(\Gamma_{sj}^m|(D)^m)<\epsilon$ then the branch is suspended. The next set of prior probabilities can now be derived and the cycle continues.

These results are derived using the following independence assumptions:

$$P(a_i^k|a_j^m,D^m) = P(a_i^k|a_j^m) \quad \text{and} \quad P(D^m|a_j^m,D^k) = P(D^m|a_j^m)$$

which decouple the data at different stages.

Figure 4 shows successive likelihoods, entered by hand for a (rather contrived) illustration using the grammar in figure 1. At the end the two viable sentences (with probabilities) are

"pn tv det n pron tv pn" (0.897)
"det n pron tv pn tv pn" (0.103)

Notice that the string which maximises the likelihood at each stage,

"pn tv pron tv pron tv pn"

might correspond to a line of poetry but is not a sentence in the language.

The graph-structured stack approach of Tomita [6] is used for non-deterministic input. Each path through the stack graph corresponds to one or more partial sentences and the probability $P(\Gamma_s^m|(D)^m)$ has to be associated with each partial sentence Γ_s^m.

3.2 *Entropy of the Partial Sentences*

Despite the pruning the number of partial sentences maintained by the parser tends to grow with the length of input. It seems sensible to base the measure of complexity upon the probabilities of the sentences rather than their number, and the obvious measure is the entropy of the distribution. The discussion here will assume that the proliferation of sentences is caused by input uncertainty rather than by action conflicts. This is likely to be the dominant factor in speech applications.

The sentence entropy $\mathcal{H}_S^m$ is defined as

$$\mathcal{H}_S^m = -\sum_{s,j} P(\Gamma_{sj}^m|(D)^m) \log P(\Gamma_{sj}^m|(D)^m)$$

where natural logarithms are used. A related measure called "perplexity" [12], defined as

$$\mathcal{P}_S^m = \exp(\mathcal{H}_S^m)$$

is the equivalent (in entropy) number of equally-likely sentences. Substituting for $P(\Gamma_{sj}^m|(D)^m)$ from equation (1) leads to

$$\mathcal{H}_S^m = -\sum_j P(a_j^m|(D)^m)\left[\log P(a_j^m|(D)^m) - h_j^m \right]$$

where

$$h_j^m = -\sum_s P(\Gamma_s^{m-1}|a_j^m,(D)^{m-1}) \log P(\Gamma_s^{m-1}|a_j^m,(D)^{m-1})$$

is the entropy contributed by the sentences at stage m-1 predicting word a_j^m. The quantities h_j^m can be evaluated with the prior probabilities.

It can be shown that the sentence entropy has an upper bound as a function of the likelihoods:

$$\mathcal{H}_S^m \leq \log \sum_j \exp(h_j^m)$$

withequality when $P(D^m|a_j^m) \propto \dfrac{\exp(h_j^m)}{P(a_j^m|(D)^{m-1})}$.

The constant of proportionality does not matter. Figure 5(a) shows this

upper bound for the grammar in figure 1, and it can be seen that the perplexity is equivalent to 35 equally-likely sentences after 10 words.

The upper bound is very pessimistic because it ignores the discriminative power of the pattern matcher. This could be measured in various ways but it is convenient to define a "likelihood entropy" $\mathcal{H}_L^m$ as

$$\mathcal{H}_L^m = -\sum_j \frac{P(D^m|a_j^m)}{\sum_i P(D^m|a_i^m)} \log \frac{P(D^m|a_j^m)}{\sum_i P(D^m|a_i^m)}$$

and the "likelihood perplexity" is given by $\mathcal{P}_L^m = \exp(\mathcal{H}_L^m)$.

The maximum sentence entropy subject to a fixed likelihood entropy can be found by simulation. Sets of random likelihoods with a given entropy can be generated from sets of independent uniform random numbers by raising these to an appropriate power. Permuting these so as to maximise the sentence entropy greatly reduces the number of sample runs needed to get a good result. These likelihoods are then fed into the parser and the procedure repeated to simulate the recognition process. The sentence entropy is maximised over a number of such runs.

The likelihoods which produce the upper bound line shown in figure 5(a) have a perplexity which is approximately constant at 6.6. This line is reproduced almost exactly by the above simulation procedure, using a fixed $\mathcal{P}_L$ of 6.6 with 30 sample runs.

The simulation method is easily adapted to compute the average sentence entropy over the sample runs. For this it is preferable to average the entropy and then convert to a perplexity rather than average the measured perplexity values. This process provides an indication of how the parser will perform in a typical case, assuming a fixed likelihood perplexity as a parameter (although this could be varied from stage to stage if required).

Figure 5(a) shows how the average compares with the maximum for a fixed $\mathcal{P}_L$ of 6.6, and how the sentence perplexity is reduced when the likelihoods are progressively more constrained ($\mathcal{P}_L$ = 5.0, 3.0 and 2.0).

3.3 *Comparison with Inferred Markov Model*

Markov models have some advantages over grammar models for speech recognition in flexibility and ease of use but a major disadvantage is their limited memory of past events. For an extended utterance the number of possible sentences compatible with a Markov model may be much greater than for a grammar model, for the same data. Demonstrating this in the present context requires the derivation of a first-order Markov model from a probabilistic grammar [13].

The uncertainty algorithm of section 3.1 will operate largely unchanged with the prior probabilities obtained from the transition probabilities rather than from the LR parser. Figure 5(b) contains results corresponding to those in (a), for the Markov model inferred from the grammar in figure 1. The upper bound reaches 409 after 10 words, for a likelihood perplexity of approximately 6.3, reducing to 37 for the average (after 30 sample runs). This falls with the likelihood perplexity but is higher than for the grammar model. The sentence perplexity for the grammar is twice that for the inferred Markov model after from six to nine words depending on $\mathcal{P}_L$. This comparison is reproduced for other grammars considered.

References

1. S E Levinson, L R Rabiner and M M Sondhi, "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition", BSTJ vol 62, pp1035-1074, 1983.

2. R Garside, G Leech and G Sampson (eds), "The Computational Analysis of English, a Corpus-Based Approach", Longman, 1987.

3. H Bourland and C J Wellekens, "Speech Pattern Discrimination and Multilayer Perceptrons", Computer Speech and Language, vol 3, pp1-19, 1989.

4. A V Aho, R Sethi and J D Ullman, "Compilers: Principles, Techniques and Tools", Addison-Wesley, 1985.

5. N P Chapman, "LR Parsing, Theory and Practice", Cambridge University Press, 1987.

6. M Tomita, "Efficient Parsing for Natural Language", Kluwer Academic Publishers, 1986.

7. J H Wright and E N Wrigley, "Linguistic Control in Speech Recognition", Proceedings of the 7th FASE Symposium, pp545-552, 1988.

8. P Suppes, "Probabilistic Grammars for Natural Languages", Synthese, vol 22, pp95-116, 1968.

9. W J M Levelt, "Formal Grammars in Linguistics and Psycholinguistics, volume 1", Mouton, 1974.

10. C S Wetherall, "Probabilistic Languages: A Review and Some Open Questions", Computing Surveys vol 12, pp361-379, 1980.

11. S M Shieber, "Sentence Disambiguation by a Shift-Reduce Parsing Technique", Proc. 21st Annual Meeting of Assoc. for Comp. Linguistics, pp113-118, 1983.

12. L R Bahl, J Jelinek and R L Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol PAMI-5, pp179-190, 1983.

13. J H Wright, "Linguistic Modelling for Application in Speech Recognition", Proceedings of the 7th FASE Symposium, pp391-398, 1988.

(1) S → NP VP, 1.0	(5) REL → pron VP, 0.3
(2) NP → pn, 0.4	(6) VP → iv, 0.5
(3) NP → det n REL, 0.6	(7) VP → tv NP, 0.5
(4) REL → λ, 0.7	

Figure 1: A simple probabilistic grammar.

STATE	ACTION							GOTO			
	pn	det	n	pron	iv	tv	$	S	NP	REL	VP
0	sr2 0.4	s1 0.6						s2	s3		
1			s4 1.0								
2							acc 1.0				
3					sr6 0.5	s5 0.5					sr1
4				s6 0.3	r4 ⟵	r4 0.7	r4 ⟶			sr3	
5	sr2 0.4	s1 0.6							sr7		
6					sr6 0.5	s5 0.5					sr5

Figure 2: SLR and LALR parsing tables for the grammar in figure 1.

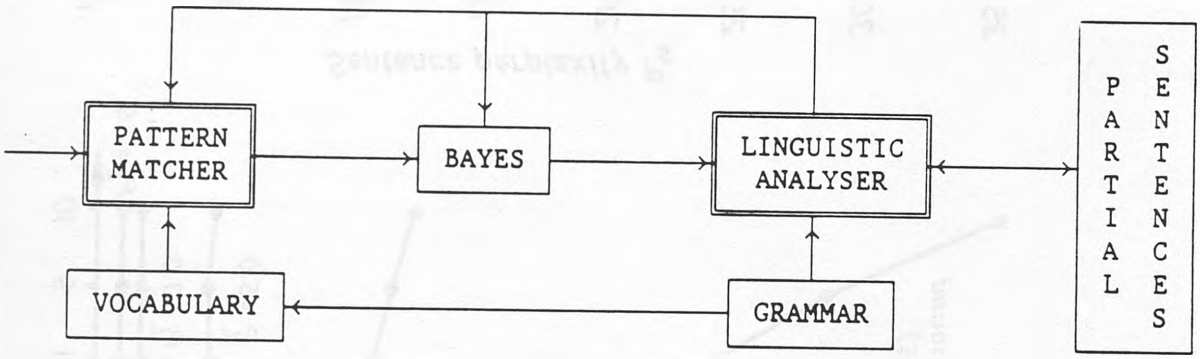

Figure 3: Linguistic control block diagram for speech recognition.

TERMINAL	STAGE (m)							
	1	2	3	4	5	6	7	8
pn	0.9			0.3	0.4		0.9	
det	0.2		0.4					
n		0.2		0.5				
pron			0.8		0.7			
iv								
tv		0.8	0.1	0.9		0.8		
$								1.0

Figure 4: Likelihoods for illustration of uncertainty algorithm.

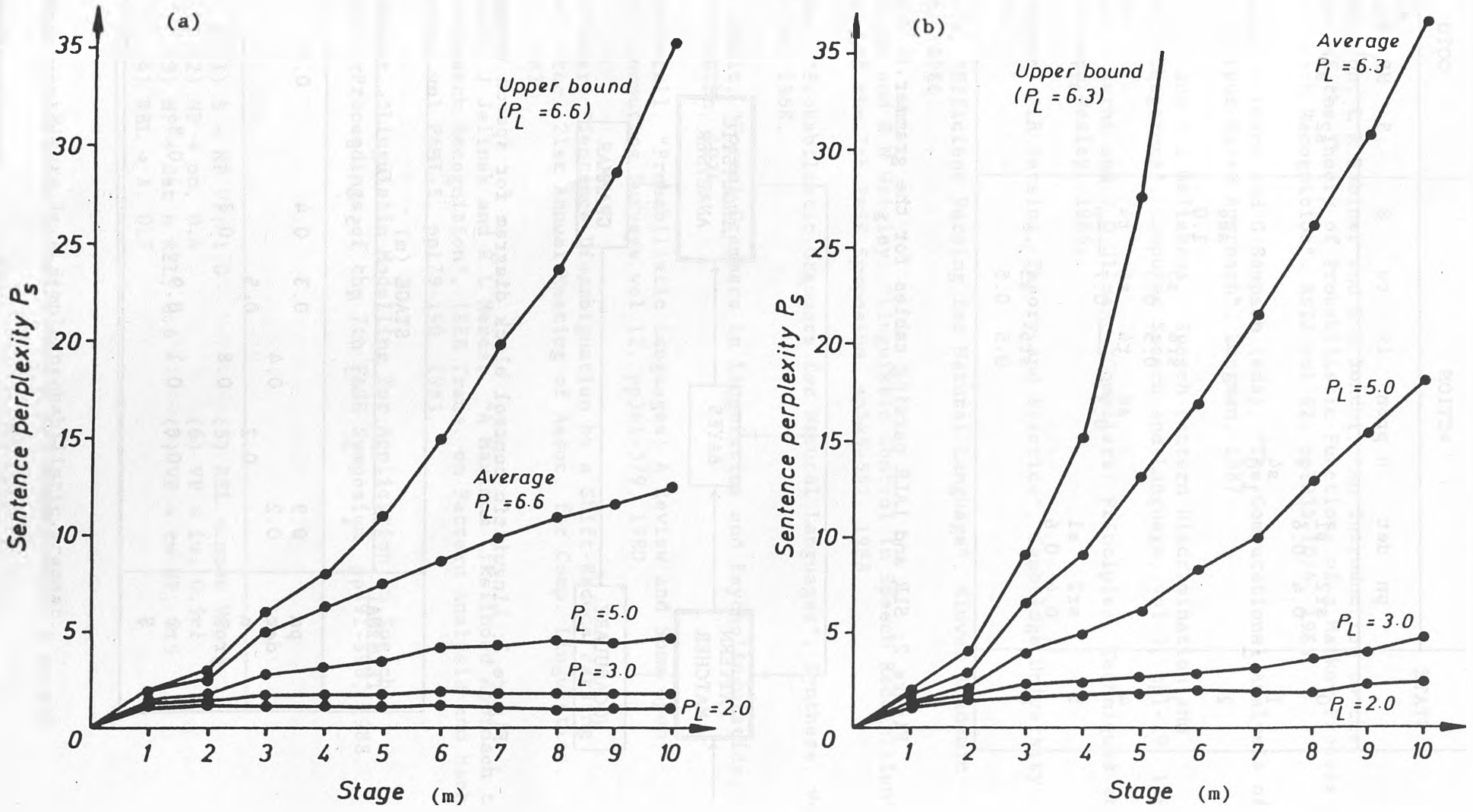

Figure 5: Sentence perplexities for (a) grammar, (b) Markov model.