K-TLSS(S) LANGUAGE MODELS FOR SPEECH RECOGNITION *

G. Bordel *A. Varona* *M. I. Torres*
Dpto. Electricidad y Electrónica
Universidad del País Vasco/Euskal Herriko Univertsitatea (UPV/EHU)
Bilbao, Spain
german@we.lc.ehu.es

ABSTRACT

The class of *K-Testable Languages in the Strict Sense* (K-TLSS) is a subclass of regular languages. Previous works demonstrate that stochastic K-TLSS language models describe the same probability distribution as N-gram models, and that smoothing techniques can be efficiently applied (Back-off like methods). Once we have a set of k-TLSS models ($k = 1 \dots K$) and a smoothing technique that specifically fits in them, here we propose an integration into a unique self-contained model (the K-TLSS(S)) which embeds the smoothing within the topology allowing extremely simple parsing procedures. To build this model we designed a more general syntactic mechanism that we call Stochastic Deterministic Finite State Automaton with Recursive Transitions. The topology of the new models (K-TLSS(S)) allows an easy pruning procedure. Pruned K-TLSS(S) models give probability distributions that are equivalent to *Variable length N-gram* models. Experimental results gave as a conclusion that the effect of a small pruning is always positive.

1. INTRODUCTION.

In 1991, talking about Language Modeling, Jelinek said: "...after all the solid progress in speech recognition, the trigram model remains fundamental" [1]. Basically, the validity of this statement continues today, so while the search for new improved methods persists, it would be worth it to make advances in the application of the N-gram model.

In this way, we paid attention to the fact that the probability distribution given by an N-gram model is strictly equivalent to the distribution determined by a stochastic grammar for a certain subclass of regular languages called *K-Testable Languages in the Strict Sense* (K-TLSS)[2]. So, the benefits of the well structured computational framework of Formal Language Theory can be applied instead of the classical N-gram procedures. In this way, an automaton can be inferred from the training set for a given value of K [3] (K stands for the same meaning as N in N-gram). In section 2 this aspect is formalized.

Nevertheless, the application of an automatically learned Language Model to a Speech Recognition task requires a solution to the "lack of samples" problem [4]. A way to give a solution to this problem for the proposed syntactic approach consists of the translation of the solutions developed for the N-gram approach. In [5] this issue is studied and a smoothing technique called "syntactic Back-off" is proposed. The origin of this technique is the classical Back-off introduced for N-grams in [6]. The mathematical expression for this smoothing is shown in section 3.

Once we have different k-TLSS models ($k = 1 \dots K$) and a smoothing technique that specifically fits in them, here we propose an integration into a unique self-contained model (the K-TLSS(S)). To build this model we designed a syntactic mechanism (the *Stochastic Deterministic Finite State Automaton with Recursive Transitions* - SDFSART) which is explained in section 4 and applied to construct the K-TLSS(S) models in section 5.

Section 6 explains how this new method allows an easy pruning step as a natural part of the construction of the model. The probability distribution provided by these models issimilar to that of the *Variable length N-grams* [7]. Some experimental results were obtained with these models revealing that a moderate pruning should always be done.

2. K-GRAM MODEL AND K-TLSS LANGUAGES.

The SDFSA for a K-TLSS is described by the quintuple $(\Sigma, Q^K, \delta^K, q_0, q_f)$ where, in terms of word chains (N-grams):

- Σ is the vocabulary inferred from the sample.
- Q^K is a set of states. Each state represents a word chain of length up to $K-1$. There is one state in Q^K for each word chain shorter than $K-1$ starting the sentences of the training text (including the null string). For each word chain of length $K-1$ in the whole training text there is also one state in Q^K.
- δ^K is a stochastic transition function ($\delta^K : \mathcal{D} \rightarrow Q^K \times [0,1]$) where $\mathcal{D} \subseteq Q^K \times (\Sigma \cup \{\$\})$ and \$ is an (internal) symbol. Each word chain shorter than K starting a sentence in the training text $\omega_{i-k+1} \dots \omega_{i-1} \equiv \omega_{i-k+1}^{i-1}$ defines a value for δ^K:

$$\delta^K(\omega_{i-k+1}^{i-1}, \omega_i) = (\omega_{i-k+1}^{i}, \hat{P}(\omega_i | \omega_{i-k+1}^{i-1})) \quad k < K$$

where $\hat{P}(\omega|\Omega)$ is the estimated probability for a word ω to appear after the string Ω.

*This work has been partially supported by the CICYT under project TIC-95-0884-C04-03. The authors would like to thank especially Enrique Vidal from Universidad Politécnica de Valencia for much helpful advice.

Each word chain of length K in the whole text also defines a value for δ^K:

$$\delta^K(\omega_{i-K+1}^{i-1}, \omega_i) = (\omega_{i-K+2}^{i}, \hat{P}(\omega_i|\omega_{i-k+1}^{i-1}))$$

Finally, for each state whose associated word chain ends a sentence in the training text:

$$\delta^K(\omega_{i-k+1}^{i-1}, \$) = (q_f, \hat{P}(\$|\omega_{i-k+1}^{i-1})) \qquad k \leq K$$

In this case, $\hat{P}(\$|\Omega)$ is the estimated probability for the string Ω being at the end of a sentence.

- q_0 is the initial state. This is the state associated to the null string.
- q_f is the final state.

3. SMOOTHING THE K-TLSS PROBABILITY DISTRIBUTION.

In [5] a back off smoothing method for K-TLSS models is proposed and a simple mechanism to implement it as part of the training procedure is developed.

Each state of the K-TLSS automaton, q, has an associate alphabet, Σ_q, formed by all the words seen when it was at q. The final equation smoothes the probabilities based on a lower level model (k-1)-TLSS in the following manner:

$$\hat{P}(\omega|q) = \begin{cases} \frac{C(\omega|q)}{C(q)+|\Sigma_q|} & if \omega \in \Sigma_q \\ \frac{|\Sigma_q|}{C(q)+|\Sigma_q|} \frac{\hat{P}(\omega|q^*)}{1-\sum_{\forall\omega' \in \Sigma_q} \hat{P}(\omega'|q^*)} & if \omega \notin \Sigma_q \end{cases}$$

where $C(\omega/q)$ is the counter for ω in the state q, $C(q)$ is the total count for the state $(\sum_{\forall\omega\in\Sigma_q} C(\omega|q))$, and $|\Sigma_q|$ is the number of words in Σ_q. $\hat{P}(\omega_i|q^*)$ is the estimated probability given by the (k-1)-TLSS model to the same situation (that is, if $q \equiv \omega_{i-k+1}^{i-1}$ then $q^* \equiv \omega_{i-k+2}^{i-1}$).

4. THE SDFSART: A SYNTACTIC MACHINE.

The previous scheme allowed the parsing of new sentences based on a set of K automata that must be run in parallel. It would be better to have only one automata whose transition function domain were the whole $\mathcal{D} \equiv Q^K \times (\Sigma \cup \{\$\})$. But this implies the expansion of the learned δ^K to all the words at each state applying the smoothing function, which is clearly prohibitive in terms of spatial efficiency.

A new kind of automata can be defined to simulate a complete δ^K function (without the expansion) by means of a recursive behaviour. That is what we call SDFSART (Figure 1).

The SDFSART is described by the quintuple $(\Sigma, Q^*, \delta R, q_0, q_f)$, where Σ, q_0, and q_f are the same defined in section 2, Q^* is the union of all the sets of states Q^k of the separate k-TLSS automata, and δR is the recursive transition function explained below:

$$\delta R \equiv (\delta R_s, \delta R_p) : Q^* \times (\Sigma \cup \{\$\}) \rightarrow Q^* \times [0,1]$$

that is:

$$\begin{aligned} \delta R_s &: Q^* \times (\Sigma \cup \{\$\}) \rightarrow Q^* \\ \delta R_p &: Q^* \times (\Sigma \cup \{\$\}) \rightarrow Q^* \times [0,1] \end{aligned}$$

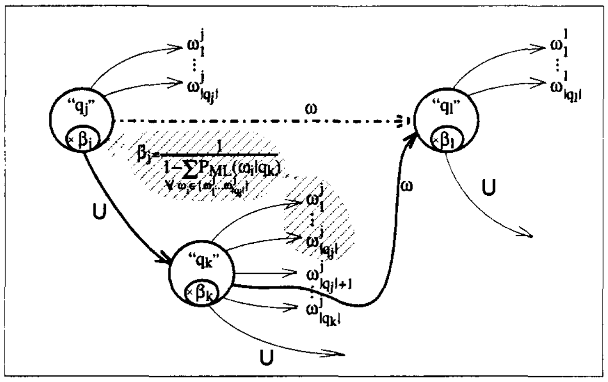

Figure 1. SDFSART models are automata where transitions can go through some states. When the word ω appears being at q_j the state q_l is reached by means of crossing q_k. This behaviour is due to the transition function (δR) which is composed of an explicit transition function (δX) and a recursive strategy.

The definition of δR is (an implementation of this function is shown in Figure 2):

$$\delta R(q,\omega) = \begin{cases} \delta X(q,\omega) & if \omega \in \Sigma_q \\ (\delta R_s(p,\omega), \delta X_p(q,\mathcal{U})\delta R_p(p,\omega)) & if \omega \notin \Sigma_q \end{cases}$$

$$\text{with } p \equiv \delta X_s(q,\mathcal{U})$$

which is based on δX, an 'extension' of the δ^k (k=1..K) functions given by the union of all these δ^k and the addition of one (internal) extra symbol to the alphabet axis of the dominion:

$$\delta X \equiv (\delta X_s, \delta X_p) : Q^* \times (\Sigma \cup \{\$\} \cup \{\mathcal{U}\}) \rightarrow Q^* \times [0,1]$$

```
function $\delta R$(state,word):stateandprob
var output, tmp :stateandprob
begin
if $\exists$  $\delta X$(state,word)
  then return $\delta X$(state,word)
  else
    begin
    tmp=$\delta X$(state,$\mathcal{U}$)
    output=$\delta R$(tmp.state,word)
    output.prob=tmp.prob $\times$ output.prob
    return output
    end
end
```

Figure 2. The recursive function δR is based on δX, which is a direct data extraction from a 'sparse matrix like' structure. The recursive function is shown for the sake of clarity, but a more efficient non-recursive function is clearly straightforward obtained from this one.

The values given to $\delta X_s(q,\mathcal{U})$ allow for a word w not seen in q, the transition to a different sub-model (to the state p).

The function δX can be seen like a sparse matrix, and stored in a very efficient way related to this kind of structures. Moreover the access can be optimized attending to the values of the probability part of the function.

The SDFSART must assure that, for each state and each word in $\Sigma \cup \{\$\}$, there is a destination state and that the stochastic condition meets:

$$\sum_{\forall \omega \in (\Sigma \cup \{\$\})} \delta X_p(q,\omega) = 1 \qquad \forall q \in Q^*$$

5. K-TLSS(SMOOTHED) AS A SDFSART.

For the smoothing formula in section 3, is easily proved that the stochastic condition is fulfilled when:

$$\delta X_p(q,\mathcal{U}) = \frac{|\Sigma_q|}{C(q)+|\Sigma_q|} \frac{1}{1-\sum_{\forall \omega' \in \Sigma_q} \hat{P}(\omega'|q^*)}$$

To assure that for each state and each word in $\Sigma \cup \{\$\}$ there is a destination state, some properties can be established. Nevertheless, we are interested in the application of the SDFSART to solve the integration of our K-TLSS into a unique smoothed model (the K-TLSS(S)), and the procedure developed meets this requirement by construction. This procedure starts building the K levels trie whose level l is formed by nodes associated to word chains of length l (see Figure 3(a)). After that, as a second step, the starting sequences are separated obtaining two tries; one node for each non-leaf node is added; and one probability is associated to each node attending to the chosen smoothing formula. As a third step, one transition for every node in the whole structure is added transforming the original trie structure into a graph with the topology of the K-TLSS(S) (Figure 3(b)).

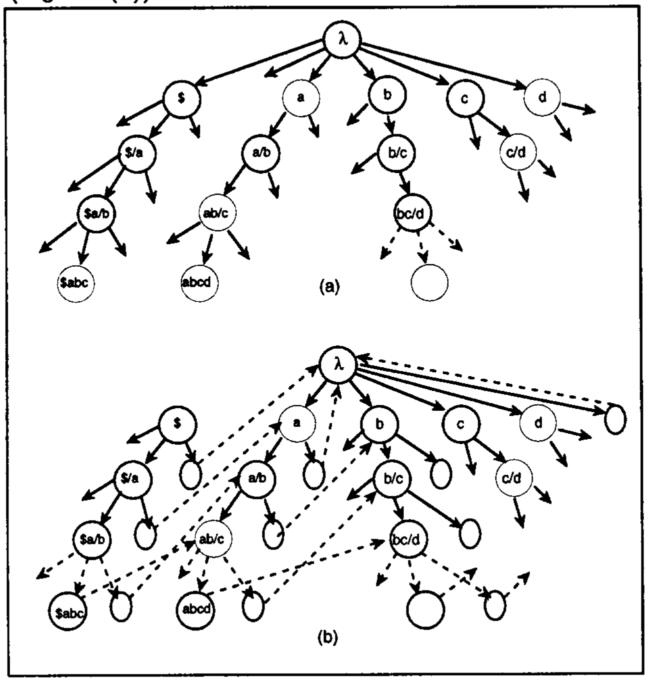

Figure 3. To build the K-TLSS(S), a trie is extracted from the sample text (a) and, after that, a process is applied to transform it into the adequate topology.

The K-TLSS(S) topology matches with the one reported, for the bigram level, in [8], and with the posterior generalization given in [9]. In the latter work, allowing null-transitions, it is always possible to reach one state assigning a probability for the next word. So, it is a non-deterministic automaton giving an alternative probability distribution to the N-gram model. The SDFSART provides the mechanism to perform a deterministic analysis giving exactly the same results as the N-gram model in a compact, robust and efficient manner.

6. PRUNING THE K-TLSS(S) MODELS. EXPERIMENTAL RESULTS.

The K-TLSS(S) has been proven to present the same performance as the separated K-TLSS models in terms of Perplexity for a test set. Effectively, the results presented in [5] applying the back-off in section 3 were reproduced with this model.

```
procedure prune_KTLSSS (KTLSSS,threshold)
var actual_node: ↑node
begin
∀ actual_node of the trie proceeding in width
  if not erased(actual_node) and
     probability(actual_node)<threshold
     then
     mark_erased_subtrie(actual_node)
pack_KTLSSS(KTLSSS)
end
```

Figure 4. The pruning procedure (Proc D in Figure 5).

Nevertheless, one of the strengths of this model is that the pruning can be easily performed, with complete disregard for the automaton structure, like the pruning of a tree (see Figure 4), if the third step of the construction procedure is re-applied (see Figure 5). Some pruning experiments gave as a result that a small pruning is always positive for the performance of the model. So, to build a K-TLSS(S), the pruning procedure must always be applied.

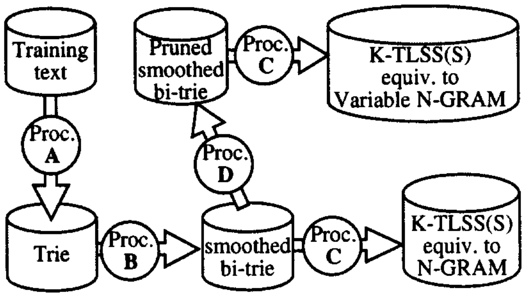

Figure 5. The building process for a K-TLSS(S) should include a pruning step on the trie (proc D) prior to the trie_to_automaton transformation (proc C).

The pruning applied consisted of eliminating those states with a probability under a certain threshold. The probability of a state is efficiently obtained applying through the trie the recursive expression:

$$\begin{aligned}\hat{P}(q_r) &= \hat{P}(\omega_{i-k+1}^{i-1}) = \\ &= \hat{P}(\omega_{i-1}|\omega_{i-k+1}^{i-2}) \times \hat{P}(\omega_{i-k+1}^{i-2}) = \\ &= \delta R_p(q_s,\omega_{i-1})P(q_s) \quad with \quad q_s = father(q_r)\end{aligned}$$

The experiments were carried out over a corpus (BDGEO) consisting of 9150 sentences. BDGEO is a task-oriented Spanish speech corpus [10][11] with 82000 words and a vocabulary of 1284 words. It consists of a set of Natural Language (spontaneous) queries to a Spanish geographic database. This is a specific task designed to test integrated systems (acoustic, syntactic and semantic modelling) in automatic speech understanding.

In order to obtain a significant size for the test set a cross-validation technique was applied [12] selecting in each partition 9100 sentences for training purposes and 50 sentences for testing. 183 partitions were made in order to obtain an effective test set composed by all the sentences in the corpus.

To illustrate the quality improvement obtained, Table 1 and the corresponding Figure 6 present the effect of pruning the models trained from the BDGEO database. The perplexity of the non-pruned models presents the typical evolution with K: there is always a value from which the perplexity increases. This effect is less important and appears at higher values of K for the pruned models.

Table 1. Perplexity and number of parameters of the K-TLSS(S) models with and without pruning (corpus BDGEO).

	Whole models		Pruned models		Δ (%)	
K	Perp.	Param.	Perp.	Param.	Perp.	Param.
3	10.691	33654	10.691	33654	-0.00	-00.00
4	10.220	76664	10.063	61481	-1.54	-29.80
5	10.262	133653	9.9898	84661	-2.65	-36.66
6	10.495	203518	10.026	101635	-4.47	-50.06
7	10.785	279071	10.082	111730	-6.52	-59.96

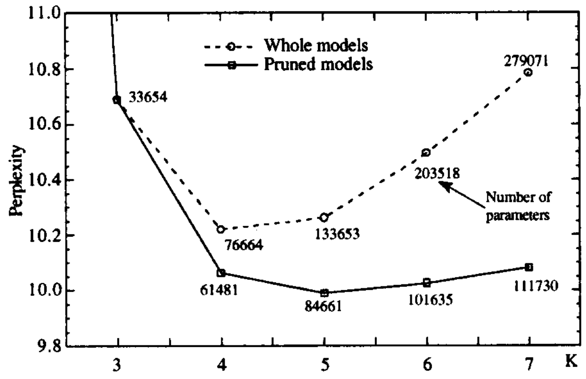

Figure 6. For any value of K, the best model is obtained after pruning. This effect allows, for a similar size, an increase in the value of K and an improvement in the performance by means of deleting some low-k states and including some more significant higher-k states.

REFERENCES

[1] F. Jelinek, "Up from trigrams: the struggle for improved language models," in *Proceedings of the Eurospeech 91*, 24 26Sept. 1991, pp. 1037–1039, Genova, Italy.

[2] E. Segarra, *Una Aproximación Inductiva a la Comprensión del Discurso Continuo*, Ph.D. thesis, DISC, Universidad Politécnica de Valencia, 1993.

[3] P. García; E. Vidal, "Inference of k-testable languages in the strict sense and application to syntactic pattern recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 9, pp. 920–925, Sept. 1990.

[4] F. Jelinek; R. L. Mercer, *Interpolated Estimation of Markov Source Parameters from Sparse Data*, pp. 381–397, North-Holland Publ. Company, Amsterdam, 1980.

[5] G. Bordel; I. Torres; E. Vidal, "Back-off smoothing in a syntactic approach to language modelling," in *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Sept. 1994, pp. 851–854, Yokohama (Japan).

[6] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. ASSP-35, no. 3, pp. 400–401, Mar. 1987.

[7] S. Deligne; F. Bimbot, "Language modeling by variable length sequences: Theoretical formulation and evaluation of multigrams," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 1995, vol. 1, pp. 169–172, Detroit.

[8] P. Placeway; R. Schwartz; P. Fung; Long Nguyen, "The estimation of powerful language models from small and large corpora," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 1993, vol. 2, pp. 33–36, Minneapolis, Minnesota.

[9] G. Riccardi; E. Bocchieri; R. Pieraccini, "Non deterministic stochastic language models for speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 1995, vol. 1, pp. 237–240, Detroit, Michigan.

[10] J. E. Diaz; A. J. Rubio; A. M. Peinado; E. Segarra; N. Prieto; F. Casacuberta, "Development of a task oriented spanish speech corpora," in *Proc. of the Eurospeech 93*, 1993, p. Addendum, Berlin, Germany.

[11] A. Moreno; D. Poch; A. Bonafonte; E. Lleida; J. Llisterri; J.B. Mariño; C. Nadeu, "Albayzin database: Design of the phonetic corpus," in *Proc. of the Eurospeech 93*, 1993, pp. 175–178, Berlin, Germany.

[12] S. J. Raudys; A. K. Jain, "Small sample effects in statistical pattern recognition: Recommendations for practitioners and open problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 252–263, 1991.