Computer Speech and Language (2001) **15**, 127–149
doi:10.1006/csla.2001.0162
Available online at http://www.idealibrary.com on IDEAL®

k-TSS language models in speech recognition systems

I. Torres and A. Varona

Departamento de Electricidad y Electrónica. Facultad de Ciencias UPV/EHU, Apdo 644-48080 Bilbao, Spain†

Abstract

The aim of this work is to show the ability of stochastic regular grammars to generate accurate language models which can be well integrated, allocated and handled in a continuous speech recognition system. For this purpose, a syntactic version of the well-known *n*-gram model, called *k*-testable language in the strict sense (*k*-TSS), is used. The complete definition of a *k*-TSS stochastic finite state automaton is provided in the paper. One of the difficulties arising in representing a language model through a stochastic finite state network is that the recursive schema involved in the smoothing procedure must be adopted in the finite state formalism to achieve an efficient implementation of the backing-off mechanism. The use of the syntactic back-off smoothing technique applied to *k*-TSS language modelling allowed us to obtain a self-contained smoothed model integrating several *k*-TSS automata in a unique smoothed and integrated model, which is also fully defined in the paper. The proposed formulation leads to a very compact representation of the model parameters learned at training time: probability distribution and model structure. The dynamic expansion of the structure at decoding time allows an efficient integration in a continuous speech recognition system using a one-step decoding procedure.

An experimental evaluation of the proposed formulation was carried out on two Spanish corpora. These experiments showed that regular grammars generate accurate language models (*k*-TSS) that can be efficiently represented and managed in real speech recognition systems, even for high values of *k*, leading to very good system performance.

1. Introduction

Continuous speech recognition (CSR) systems transcribe speech signals into sequences of linguistic units $\Omega \equiv \omega_1\omega_2 \ldots \omega_{|\Omega|}$, usually words, from some previously established finite vocabulary $\Sigma = \{\omega_j\}$, $j = 1 \ldots |\Sigma|$. Given a sequence of acoustic measurements $A = a_1 a_2 \ldots a_{|A|}$, the recognizer must find the sequence of linguistic units that maximizes the probability that the sequence Ω was spoken, given that the acoustic sequence A was observed (Jelinek, 1985). Thus, the following optimization problem has to be solved:

$$\hat{\Omega} = \arg\max_{\Omega} P(\Omega/A). \tag{1}$$

†E-mail: {manes,amparo}@we.lc.ehu.es

0885–2308/01/020127 + 23 $35.00/0

Using the well-known Bayes' formula, $P(\Omega/A)$ can be rewritten as

$$\hat{\Omega} = \arg\max_{\Omega} P(\Omega) P(A/\Omega) \tag{2}$$

where $P(\Omega)$ is the probability that the word sequence Ω will be uttered and $P(A/\Omega)$ is the probability of the sequence of acoustic observations A for a given sequence of words Ω. Probabilities $P(A/\Omega)$ are represented by an acoustic model, usually hidden Markov models (HMM). The *a priori* probabilities $P(\Omega)$ are given by a language model (LM).

The most successful parsing method for dealing with Equation (2) is the time-synchronous decoding achieved by the Viterbi algorithm (Rabiner & Juang, 1993). This algorithm finds the most likely path, i.e. the optimal state sequence through the trellis (Forney, 1973). This path will lead through a sequence of word models that do not necessarily correspond to the string $\hat{\Omega}$ defined in Equation (2) (Jelinek, 1999). A second approach is the stack decoding algorithm. This finds the best sequence of words defined in (2) by exploring the tree of linguistic hypotheses (Jelinek, 1976, 1999). However, the problems arising in defining adequate heuristic functions to prune all the word hypotheses make this algorithm difficult to use for speech recognition applications (Bahl, de Genaro, Gopalakrishnan & Mercer, 1993; Riccardi, Pieraccini & Bocchieri, 1996; Jelinek, 1999).

Multipass searches have also been proposed. A simple language model (bigram) is used in a first decoding step to obtain a pruned lattice of word hypotheses. More accurate language models can then be applied to the reduced graph to construct a new trellis. This strategy can be extended for better, and more complex, language models (Schwartz & Austin, 1991). However, the pruning strategies can affect the search for the best word sequence (Bahl *et al.*, 1993; Riccardi *et al.*, 1996; Jelinek, 1999).

The language model used determines the complexity of the final search network (Jelinek, 1999). Thus, many authors share the objective of obtaining accurate language models that can be well integrated into the maximization procedure in Equation (2), and can be handled in a one-step decoding procedure (Placeway, Schwartz, Fung & Nguyen, 1993; Riccardi *et al.*, 1996; Bonafonte & Mariño, 1998; Llorens, 2000). This is also one of the main goals of our work (Bordel, Torres & Vidal, 1994; Varona & Torres, 1999; Torres & Varona, 2000; Varona, 2000).

This work focuses on language modelling. Statistical methods have been used extensively to generate the LM. They are based on the estimation of the probability of observing the $n-1$ preceding lexical units (n-gram models): $P(\omega_i/\omega_1 \ldots \omega_{n-1})$. The number of probabilities to be taken into account is an exponential function of n and therefore very large training corpora are needed for their correct estimation. Thus, in practice, the use of this kind of model is restricted to low values of n, typically bigrams and trigrams in large-vocabulary recognition tasks. In such a case, this formulation is only able to represent very local constraints and, therefore, does not adequately model the inherent redundancy of language.

The use of stochastic automata to represent statistical language models (n-grams) has recently been proposed (Placeway *et al.*, 1993; Zhao, Kenny, Labute & O'Shoughnessy, 1993; Riccardi, Bochieri & Pieraccini, 1995; Riccardi *et al.*, 1996; Bonafonte & Mariño, 1998; Suzuki & Aso, 1999; Llorens, 2000) with the aim of handling accurate language models in a one-step decoding procedure. But stochastic finite state automata (SFSA) are machines which accept stochastic regular languages under a syntactic approach. Thus, these proposals use syntactic tools (SFSA) to represent statistical models.

The use of a grammar formalism in language modelling presents several advantages (Vidal, Casacuberta & García, 1995; Pereira & Riley, 1997; Aubert, 2000) since language constraints can be better modelled by using a syntactic approach. Regular languages can account for *local*

or short-term constraints (such as *n*-grams) and also for more *global* or long-term constraints that often underlie in natural languages. Context free languages are much more powerful than regular languages. They can properly account for subtle language constraints and admit very compact representations (Fu, 1974; Hopcroft & Ullman, 1979; Vidal *et al.*, 1995; Dupont & Miclet, 1998). Consider the following utterances in a local telephone exchange task (Vidal *et al.*, 1995):

please, I wanted to talk to Mr. X
may I speak with Mr. Y, please
please, I would like to talk to Mr. Z, please

A very simple regular grammar can be considered to generate a language model that allows the first and second sentences but not the third. The regular language can be expressed as $(please\ G\ \lambda) \cup (\lambda\ G\ please)$, where the whole grammar G represents the language of phone requests and λ the *null* string. In this example the grammar G needs to be replicated. A more compact representation of the LM that avoids the grammar duplication can be obtained through a context-free grammar (Vidal *et al.*, 1995). Some formalisms based on regular grammars and context free grammars have been used in LM (Jelinek, Lafferty & Mercer, 1992; Vidal *et al.*, 1995; Dupont & Miclet, 1998; Caseiro & Trancoso, 2000; Mohri, Pereira & Riley, 2000).

The use of stochastic regular grammars to generate LM leads to the use of a corresponding SFSA at decoding time. In this framework, the probability $\hat{P}(\Omega)$ is computed as the probability of the string Ω being accepted by the SFSA. A sequence of states is associated with each string of words for a deterministic SFSA. Thus, any decoding algorithm based on a network search is particularly effective for working with a SFSA and can potentially improve its efficiency. In particular, speech recognition is performed by searching for the word sequence that maximizes the probability that the sequence Ω was spoken according to Equation (2). The Viterbi algorithm searches for the "best path" in a network. When the LM has been generated by a stochastic regular grammar such a network is composed of a set of two level finite state networks: the SFSA representing the LM guiding the search procedure on the first level and a chain of stochastic acoustic models, usually HMMs, replacing each word on the second level. Thus, the maximization problem represented in Equation (2) can be obtained in a one-step decoding procedure. Moreover, additional structures such as category *n*-grams (Brown, deSouza, Mercer, dellaPietra & Lai, 1992; Niesler & Woodland, 1999), network representing crossword constraints (Mohri *et al.*, 2000), etc., can be straightforwardly introduced into the finite state network (Aubert, 2000; Caseiro & Trancoso, 2000), increasing the efficiency of the decoder.

However, several difficulties arise in representing LM through an SFSA:

(1) A set of grammar inference techniques is required to obtain the structure and probability distributions of the SFSA from sets of training samples. Some examples can be found in literature (García & Vidal, 1990; Prieto & Vidal, 1992; Segarra, 1993). However, only restricted systems are found in CSR applications (Prieto & Vidal, 1992) due to the computational problems involved (Segarra, 1993; Llorens, 2000).
(2) Direct representation of the whole network representing the SFSA is usually prohibitive even for small vocabulary tasks (Riccardi *et al.*, 1996; Mohri *et al.*, 2000; Varona & Torres, 2000*b*).
(3) Smoothing procedures need to be developed under the syntactic approach (Bordel *et al.*, 1994; Riccardi *et al.*, 1996; Varona & Torres, 1999, 2000*a*; Llorens, 2000). The

recursive schema usually involved in the smoothing procedure must be adopted in the finite state formalism to achieve efficient implementations while obtaining adequate probability distributions.

This work deals with these problems. Its aim is to show the ability of stochastic regular grammars to generate accurate language models which can be well integrated into the full scheme given by Equation (2). For this purpose, a stochastic grammar generating a certain subclass of regular languages called k-testable language in the strict sense (k-TSS) (García & Vidal, 1990; Segarra, 1993; Bordel *et al.*, 1994; Varona & Torres, 1999, 2000*b*; Torres & Varona, 2000) is considered. In previous works (Bordel *et al.*, 1994; Varona & Torres, 1999) the use of k-TSS language models has been proposed for integration into a CSR system. Given a positive sample R^+ of strings of an unknown language, the inference algorithm to obtain a deterministic finite-state automaton that recognizes the smallest k-TSS language containing R^+ is proposed in García and Vidal (1990). The stochastic extension required to speech recognition applications is supplied in Segarra (1993). This approach represents a syntactic formalism of the well-known n-grams derived from formal languages theory (García & Vidal, 1990). In Section 2 k-TSS languages are defined. The relationship between n-grams and k-TSS is also considered in Section 2. Then the corresponding k-TSS stochastic finite state automaton (k-TSS, or n-gram, SFSA) is defined in Section 3.

Difficulties (2) and (3) are sometimes addressed together since they are related subjects. Some compact representations of the SFSA have been proposed to deal with problem (2) (Zhao *et al.*, 1993; Riccardi *et al.*, 1996; Bonafonte & Mariño, 1998). However, they lie in non-deterministic automata and as a consequence, several paths accepting the same string of words can be found in the network. The Viterbi decoding algorithm searches for the "best" sequence of states through the network. In such a case, the probability distribution obtained through the non-deterministic automaton is only an approximation to that obtained when the full deterministic SFSA is used (Zhao *et al.*, 1993; Riccardi *et al.*, 1996; Bonafonte & Mariño, 1998). Moreover, the *if...then...else* mechanism required by any smoothing technique cannot be achieved, since it does not agree with a network which accepts a sequence of words through several paths [problem (3)]. Thus, it can be considered that classical smoothing techniques have not been fully formalized under a syntactic approach (Llorens, 2000). In Sections 4–6 we deal with problems (2) and (3).

In Section 4 a syntactic back-off smoothing (Bordel *et al.*, 1994) is applied to the k-TSS SFSA modifying the probability distribution in order to consider those events not represented in the training corpus, that is, *unseen* events. The use of the syntactic back-off smoothing technique applied to k-TSS language modelling allows us to obtain a self-contained smoothed model integrating K k-TSS automata, where $k = 1, \ldots, K$. This formulation leads to a very compact representation of the model parameters learned at training time, i.e. probability distribution and model structure (Varona & Torres, 1999; Torres & Varona, 2000) (see Section 5). This compact structure allows non-deterministic parsing when used in a CSR system leading to different probability distributions. However, the smoothed k-TSS SFSA structure can be dynamically expanded at decoding time to implement the real back-off mechanism. This proposal allows an efficient use of SFSA in CSR systems while keeping the probability distributions as they were defined by smoothing techniques. It also allows the use of the time-synchronous Viterbi algorithm for parsing each new sentence. These problems and proposals are fully explained in Section 6.

An experimental evaluation of the proposed formulation was carried out on two Spanish application tasks (Section 7). These experiments showed that k-TSS language models can be

efficiently represented and managed in real CSR systems, even for high values of k, leading to very good system performances in terms of both word error rates and decoding times. Finally, some concluding remarks are presented in Section 8.

2. *k*-Testable in the strict sense languages

Let (Σ, I_k, F_k, T_k) be a four-tuple where Σ is a finite alphabet, I_k and $F_k \subseteq \bigcup_{i=1}^{k-1} \Sigma^i$ are two sets of initial and final segments, respectively, and $T_k \subset \Sigma^k$ is a set of forbidden segments of length k. A k-testable language in the strict sense $L_{k\text{-TSS}}$ is defined by a regular expression as (García & Vidal, 1990; Segarra, 1993):

$$L_{k\text{-TSS}} = \{\omega \in \Sigma^* / \omega \in I_k\Sigma^* \cap \Sigma^* F_k - \Sigma^* T_k \Sigma^*\}. \tag{3}$$

Thus, $L_{k\text{-TSS}}$ is a subset of Σ^* consisting of all the strings $\{\omega \in \Sigma^*\}$ such that:

- initial segments of ω belong to a given set I_k, which consists of strings of length up to $k-1$;
- final substrings of ω belong to a given set F_k, which consists of strings of length up to $k-1$;
- ω does not include any substring of a given set T_k, which consists of strings of length k.

k-TSS languages are a subclass of regular languages and can be inferred from a set of positive samples by an inference algorithm (García & Vidal, 1990). In such a procedure, the language inferred for a particular value of k is included in the one inferred for $k-1$: $L_{k\text{-TSS}} \subseteq L_{(k-1)\text{-TSS}} \subseteq \cdots \subseteq L_{1\text{-TSS}}$. Thus, a series of k-TSS languages can be inferred from a set of positive samples from $k = 1$ to a certain maximum value of k. This maximum value is achieved when the inferred language matches the sample strings; higher values of k would infer the same language. The 1-TSS language corresponds to the free monoid $L_{1\text{-TSS}} = \Sigma^*$.

Figure 1 shows an example of k-TSS languages inferred from a given set of positive samples $R^+ = \{abba, aaabba, bbaaa, bba\}$. The graph representing the finite network for each k-TSS language inferred is shown in the figure. Nodes are labelled by strings appearing in R^+ with length up to $k-1$. Arcs are labelled by symbols of the alphabet appearing in the training corpus at the end of the string labelling the source node. The initial state is labelled by λ; final states are represented by a double circle. The inference algorithm evaluates each sample string setting arcs and new nodes when needed (García & Vidal, 1990). In the example of Figure 1, only five different k-TSS languages can be inferred from R^+ since $L_{5\text{-TSS}}$ consists of exactly the four samples in R^+. Thus, $L_{5\text{-TSS}} \subseteq L_{4\text{-TSS}} \subseteq \cdots \subseteq (L_{1\text{-TSS}} = \Sigma^*)$.

The k-TSS language model defined above can be considered as a syntactic version of an n-gram. Moreover, it has been shown (Segarra, 1993; Dupont & Miclet, 1998) that the probability distribution obtained through an n-gram model is equivalent to the distribution obtained by a stochastic grammar generating k-TSS language, where k plays the same role as n does in n-grams. But while n-gram probability distributions are defined over fixed length strings, stochastic k-TSS languages, as with any regular language, are defined over Σ^*. In fact, special symbols are often added to n-grams to simulate the behaviour of grammars (Clarkson & Rosenfeld, 1997). Thus, the formal equivalence between n-grams and stochastic k-TSS languages is an open problem and remains to be established. N-gram models were first defined in the context of information theory (Channon & Weaver, 1981), whereas k-TSS languages come from the formal language theory and grammar inference background. N-gram models do not include any of the structural features found in k-TSS. A recent work (Llorens, 2000) presents an interesting formalization of an n-gram as a SFSA and opens up a new way

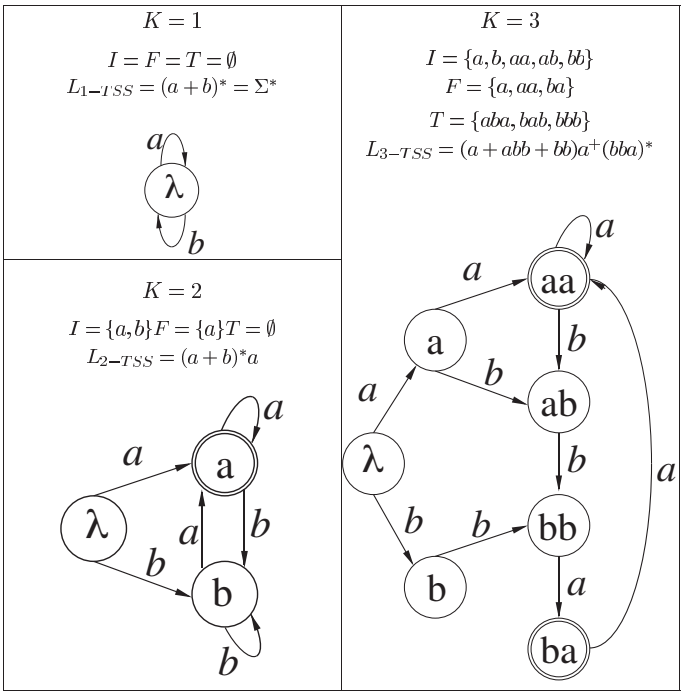

Figure 1. $L_{1\text{-TSS}}$, $L_{2\text{-TSS}}$ and $L_{3\text{-TSS}}$ inferred from a set of positive samples $R^+ = \{abba, aaabba, bbaaa, bba\}$ and an alphabet $\Sigma = \{a, b\}$. I = {initial substrings of length up to $k-1$}, F = {final substrings of length up to $k-1$} and T = {excluding substrings of length k}. Each inferred k-TSS language is included in the inferred $k-1$ language $L_{3\text{-TSS}} \subseteq L_{3\text{-TSS}} \subseteq L_{1\text{-TSS}}$. The 1-TSS language is equal to Σ^*. $L_{5\text{-TSS}}$ consists of exactly the four samples in R^+.

to analyse the formal differences between *n*-grams and *k*-TSS languages. In Section 3 the *k*-TSS SFSA is presented and some of these questions are outlined.

However, from a practical point of view, choosing *k*-TSS or *n*-grams is just a matter of representational convenience since, as mentioned above, the probability distributions obtained through the two models are equivalent (Segarra, 1993; Bordel *et al.*, 1994; Dupont & Miclet, 1998; Varona & Torres, 1999). As a consequence, the representation problems of the SFSA and the lack of syntactic smoothing[1] techniques (see Section 1) are shared by *k*-TSS and *n*-gram language models, when *n*-grams are represented through SFSA. Sections 4–6 deal with these problems.

3. The *k*-TSS stochastic finite state automaton

In the syntactic approach, stochastic *k*-TSS grammars can be used to generate language models (Bordel *et al.*, 1994; Bordel, 1996). These models need to be integrated with the stochastic acoustic models previously estimated from a set of speech training samples in a CSR system [Equation (2)]. Therefore, the corresponding *k*-TSS stochastic finite state automaton (*k*-TSS SFSA) needs to be defined and fully formalized.

[1]Syntactic smoothing ≡ smoothing technique developed to work correctly over an automaton representing a syntactic LM (Bordel *et al.*, 1994).

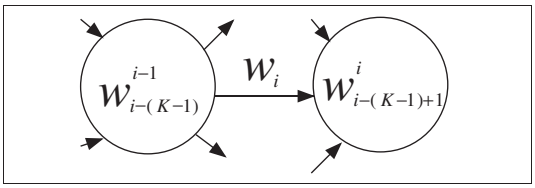

Figure 2. Two states of the automaton labelled by $\omega_{i-(k-1)}\omega_{i-(k-1)+1}\ldots\omega_{i-1}$ and $\omega_{i-(k-1)+1}\ldots\omega_{i-1}\omega_i$. When the k-gram $\omega_{i-(k-1)}\omega_{i-(k-1)+1}\ldots\omega_{i-1}\omega_i$ is observed, an outgoing transition from the first to the second state is set and labelled by ω_i.

A stochastic finite state automaton representing a k-gram model can be directly obtained from a set of training samples (García & Vidal, 1990; Segarra, 1993; Bordel, 1996). Such an automaton (k-TSS SFSA) is defined as the five-tuple $(\Sigma, Q^k, q_{0k}, F, \delta^k)$ where:

- $\Sigma = \{\omega_j\}$, $j = 1\ldots|\Sigma|$, is the vocabulary, that is the set of words appearing in the training corpus.
- Q^k is the set of states associated with the model of order k. Each state represents a string of $k-1$ words $\omega_{i-(k-1)}\ldots\omega_{i-1}$, where i stands for a generic index in any string $\omega_1\ldots\omega_i\ldots$ appearing in the training corpus. Such a state is labelled as $\omega_{i-(k-1)}^{i-1}$. A subset of Q^k represents initial substrings shorter than $k-1$ corresponding to strings of length up to $k-1$ in set I_k of the corresponding L_k-TSS (see Section 2). These states are labelled as ω_{i-j}^{i-1}, where $j = 1\ldots k-2$.
- The automaton has a unique initial state $q_0 \in Q^k$, which is labelled as λ and represents the null string.
- F is the set of final states of the automaton and represents the final substrings of length $k-1$. These substrings, along with final substrings with length up to $k-1$, belong to the set F_k of the corresponding L_k-TSS (see Section 2).
- δ^k is the transition function $\delta^k : Q^k \times \Sigma \to Q^k \times [0\ldots1]$. $\delta^k(q, \omega_i) = (q_d, P(\omega_i/q))$ defines a destination state $q_d \in Q^k$ and a probability $P(\omega_i/q) \in [0\ldots1]$ to be assigned to each element $(q, \omega_i) \in Q^k \times \Sigma$. Each transition represents a k-gram; it is labelled by its last word ω_i and connects two states labelled by $k-1$ words. Figure 2 represents two states of the automaton labelled by $\omega_{i-(k-1)}\omega_{i-(k-1)+1}\ldots\omega_{i-1}$ and $\omega_{i-(k-1)+1}\ldots\omega_{i-1}\omega_i$. When the k-gram $\omega_{i-(k-1)}\omega_{i-(k-1)+1}\ldots\omega_{i-1}\omega_i$ is observed, an outgoing transition from the first to the second state is set and labelled by ω_i; $P(\omega_i/\omega_{i-(k-1)}^{i-1})$ is the probability associated with the observed k-gram $\omega_{i-(k-1)}\omega_{i-(k-1)+1}\ldots\omega_{i-1}\omega_i$. Thus

$$\delta^k(\omega_{i-(k-1)}^{i-1}, \omega_i) = (\omega_{i-(k-1)+1}^{i}, P(\omega_i/\omega_{i-(k-1)}^{i-1})). \tag{4}$$

The stochastic constraints require that

$$\sum_{\forall\omega\in\Sigma} P(\omega/q) = 1 \qquad \forall q \in Q^k. \tag{5}$$

Under this formalism, the 1-gram involves a single state, labelled by λ, representing the *null* string, and a number of outgoing transition equal to $|\Sigma|$. The probability $P(\omega_j/\lambda)$ is the estimated probability $P(\omega_j)$ and represents the frequency of ω_j in the training corpus

$$\delta^k(\lambda, \omega_j) = (\lambda, P(\omega_j/\lambda)) = (\lambda, P(\omega_j)) \qquad \forall\omega_j \in \Sigma. \tag{6}$$

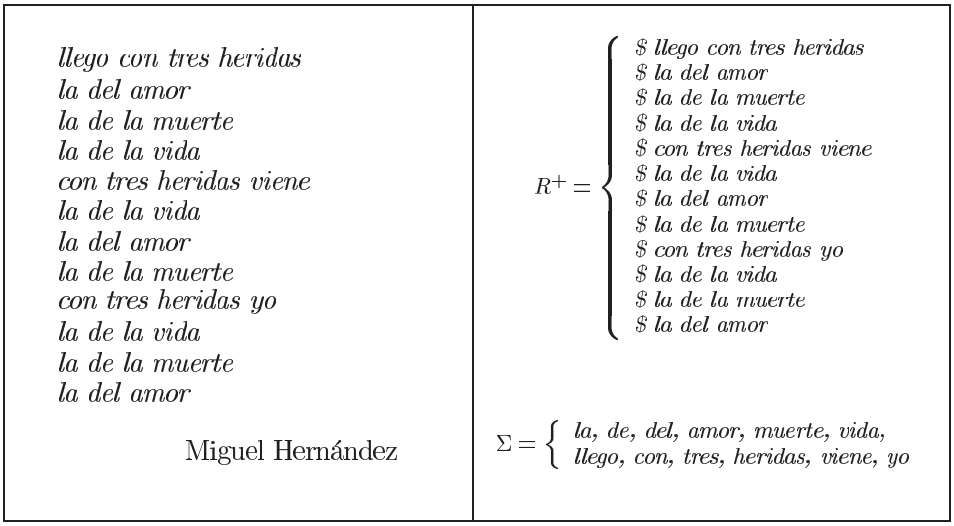

Figure 3. Selected training corpus for the k-TSS automata of Figure 4.

Examples of several k-TSS automata are shown in Figure 4. A famous poem by the Spanish poet Miguel Hernández has been selected to act as the sample text corpus. Figure 3 shows this poem, the corresponding training set R^+ and vocabulary Σ. Figure 4 shows several k-TSS automata, $k = 1 \ldots 4$, obtained from R^+ and Σ in Figure 3.

The model defined above is a deterministic, and hence unambiguous, stochastic finite state automaton (García & Vidal, 1990). Thus, the probability assigned to a sentence $\Omega \equiv \omega_1 \ldots \omega_l$ of length l, i.e. the probability of string Ω being accepted by the automaton is obtained as the product of the probabilities of the transitions used to accept Ω:

$$P(\Omega) = \prod_{i=1}^{l} P(\omega_i / \omega_{i-(k-1)}^{i-1}). \tag{7}$$

The unambiguity of the automaton also allows us to obtain a maximum likelihood estimation of the probability of each transition $\delta^k(\omega_{i-(k-1)}^{i-1}, \omega_i)$ as (Chandhuri & Booth, 1986)

$$P(\omega_i / \omega_{i-(k-1)}^{i-1}) = \frac{N(\omega_i / \omega_{i-(k-1)}^{i-1})}{\sum_{\forall \omega_j \in \Sigma} N(\omega_j / \omega_{i-(k-1)}^{i-1})} \tag{8}$$

where $N(\omega_i / \omega_{i-(k-1)}^{i-1})$ is the number of times the word ω_i appears at the end of the k-gram $\omega_{i-(k-1)} \ldots \omega_{i-1}\omega_i$, that is the count associated with the transition labelled by ω_i coming from the state labelled as $\omega_{i-(k-1)}^{i-1}$.

Such a model can only associate a probability with strings of words (k-grams) that have been observed in the samples available, i.e. seen events. Thus, large vocabulary CSR systems require a model able to associate a probability with any string of words in the vocabulary, i.e. Σ^*, and to take into account k-grams that have not been observed in the available set of data: unseen events. Consequently, a smoothing technique is needed.

$k = 1$ $k = 2$

$I = F = T = \emptyset$

$L_{1-TSS} = (a+b)^* = \Sigma^*$

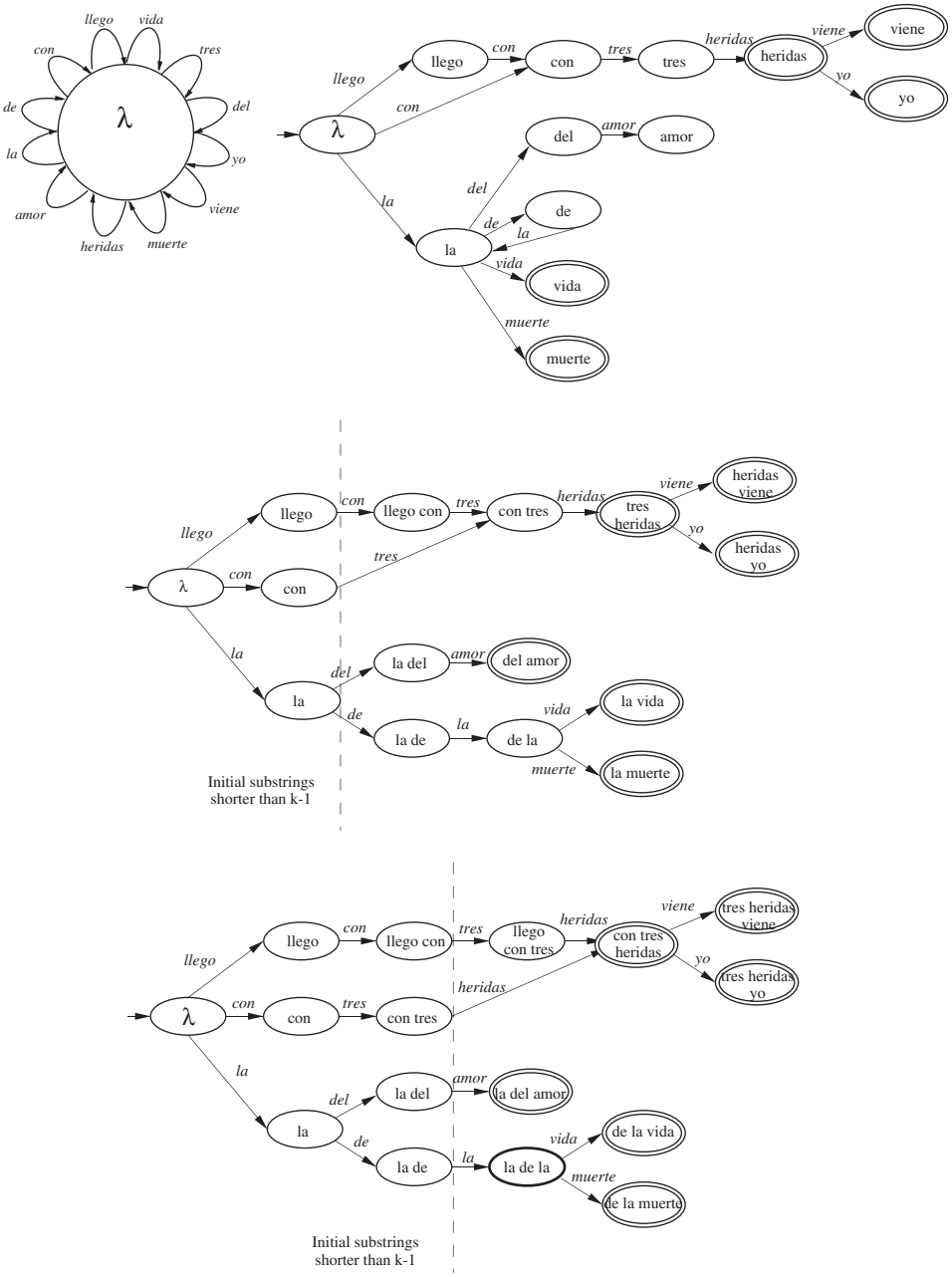

Figure 4. *k*-TSS automata, $k = 1 \ldots 4$, obtained from R^+ and Σ in Figure 3.

4. The syntactic back-off smoothing procedure

In classical smoothing techniques, a specific probability mass is reserved to be shared among unseen events, i.e. k-grams not appearing in the training corpus. In back-off smoothing (Katz, 1987) the probability to be assigned to unseen k-grams is recursively obtained from less accurate models, i.e. $k-1, \ldots, 1$. At each step of the recursion different formulations (Clarkson & Rosenfeld, 1997; Ney, Martin & Wessel, 1997) can be applied to estimate the discounted probability. In any case, the same discount is established for all k-grams having the same counts.

In previous works (Bordel *et al.*, 1994; Bordel, 1996) a syntactic back-off smoothing procedure was developed. The syntactic approach suggested a state-dependent estimation of the total discount. The symmetry principle was then locally applied to give a more accurate distribution of the probability to be assigned to unseen events (Bordel *et al.*, 1994). Under this syntactic formalism, the probability $P(\omega_i/q)$ to be associated with a transition $\delta^k(q, \omega_i)$ is estimated according to

$$P(\omega_i/q) = \begin{cases} \frac{N(\omega_i/q)}{N(q)+|\Sigma_q|} & \omega_i \in \Sigma_q \\ \frac{|\Sigma_q|}{N(q)+|\Sigma_q|} \frac{P(\omega_i/b_q)}{1-\sum\limits_{\forall \omega_i \in \Sigma_q} P(\omega_j/b_q)} & \omega_i \in (\Sigma - \Sigma_q) \end{cases} \tag{9}$$

where Σ_q is the vocabulary associated with state q, consisting of the set of words appearing after the string labelling state q in the training corpus, i.e. words labelling the set of seen outgoing transitions from state q; $N(\omega_i/q)$ is the number of times that word ω_i appears after the string labelling state q; $N(q) = \sum_{\forall \omega_i \in \Sigma_q} N(\omega/q)$; $|\Sigma_q|$ is the size of Σ_q and $P(\omega_i/b_q)$ is the estimated probability associated with the same event in the $(k-1)$-TSS model; thus, if state q is labelled as $\omega_{i-(k-1)}^{i-1}$, and $q \equiv \omega_{i-(k-1)}^{i-1} \in Q^k$, then its associated back-off state b_q is labelled as $\omega_{i-(k-1)+1}^{i-1}$, and $\omega_{i-(k-1)+1}^{i-1} \in Q^{k-1}$. In previous works (Bordel *et al.*, 1994; Varona & Torres, 2000*b*), this discounting was experimentally compared to other classical back-off methods leading to a significant decrease in test-set perplexity.

By using Equation (9) instead of Equation (8) the stochastic finite state automaton defined above is now smoothed and can associate a probability with any string of words in the vocabulary, i.e. Σ^*. However, in such a scheme a set of $|\Sigma|$ transitions, representing each word of the vocabulary, needs to be handled at each state. The smoothing function [Equation (9)] estimates $|Q^k| \times |\Sigma|$ parameters that need to be allocated a large amount of space, which is prohibitive even for small vocabulary tasks. Thus, a straightforward full network representation is not possible due to the high number of parameters to be handled. As mentioned above (Riccardi *et al.*, 1996), this is one of the difficulties arising in representing an n-gram language model through a stochastic finite state network. The syntactic formulation along with syntactic back-off smoothing can be represented in such a way that only transitions seen at training time need to be explicitly represented (Varona & Torres, 1999; Torres & Varona, 2000).

In Equation (9) the probabilities to be associated with the set of words appearing after the string labelling state q in the training corpus—the vocabulary of the state Σ_q—are explicitly estimated by calculating $N(\omega_i/q)$, $\forall \omega_i \in \Sigma_q$, and $N(q)$. The remaining $(|\Sigma| - |\Sigma_q|)$ transition probabilities corresponding to those events not represented in the training corpus are estimated according to a more general probability distribution in the $(k-1)$-TSS model. These transitions do not need to be explicitly estimated or represented at each state. The structure of the above automaton along with the back-off smoothing technique leads them to become grouped into a unique transition to a back-off state b_q. In Equation (9) the stochastic

condition should also be satisfied:

$$\sum_{\forall \omega \in \Sigma} P(\omega/q) = 1 \qquad \forall q \in Q^k. \tag{10}$$

The probability to be assigned to the transition from each state to its back-off state, $P(b_q/q)$, can then be easily estimated from Equations (9) and (10). Thus

$$P(b_q/q) = \frac{|\Sigma_q|}{N(q) + |\Sigma_q|} \frac{1}{1 - \sum\limits_{\forall \omega_j \in \Sigma_q} P(\omega_j/b_q)}. \tag{11}$$

This transition connects each state q with its back-off state b_q that represents the same event in the $(k-1)$-TSS model. Thus, if state q is labelled by $\omega_{i-(k-1)}^{i-1}$ then state b_q is labelled by $\omega_{i-(k-1)+1}^{i-1}$. The probability to be associated with each event not represented in the training corpus $P(\omega_j/q)\ \forall \omega_j \in (\Sigma - \Sigma_q)$ is estimated according to

$$P(\omega_j/q) = P(b_q/q)P(\omega_j/b_q) \qquad \forall \omega_j \in (\Sigma - \Sigma_q). \tag{12}$$

This procedure reduces the number of parameters to be handled from $|Q^k| \times |\Sigma|$ to $|Q^k| \times |\Sigma_q|$. This formulation is the application of a classical back-off smoothing schema (Katz, 1987) to a syntactic LM represented through an SFSA, where the symmetry principle is locally applied (Bordel *et al.*, 1994). The discounting in Equations (9) and (11) is analogous to the Witten–Bell discounting procedure appearing in Clarkson and Rosenfeld (1997). Alternatively, other discounting procedures have also been applied in the same syntactic back-off framework (Varona, 2000; Varona & Torres, 2000*a*).

5. The smoothed and integrated model

One of the main problems to be addressed when representing an n-gram language model through a stochastic finite state network (Riccardi *et al.*, 1996; Varona & Torres, 1999) is that the recursive schema usually involved in the smoothing procedure must be adopted in the finite state formalism to achieve efficient implementation of the back-off mechanism. The use of the syntactic back-off smoothing technique applied to k-TSS language modelling allowed us to obtain a unique model integrating the required recursive structure: a smoothed and integrated K-TSS SFSA. A smoothed K-TSS SFSA is a self-contained model that integrates K k-TSS automata, where $k = 1, \ldots, K$ in a unique automaton. Such a model is fully defined in the Appendix.

The smoothed and integrated automaton includes the definition of a set of $|\Sigma_q| + 1$ transitions associated with each state of the automaton. $|\Sigma_q|$ of these transitions are associated with k-grams, $k = 1, \ldots, K$ seen at training time [types (a) to (e) in the definition in the Appendix]. Each state of the automaton $q \in Q^K$, except the state labelled by λ, should then add a new transition to its back-off state b_q:

$$\delta^K(q, U) = (b_q, P(b_q/q)) \tag{13}$$

where U represents any unseen event associated with state q which is labelled by a word ω_j in $(\Sigma' - \Sigma_q)$. Transition $\delta^K(q, U)$ has a destination state, b_q, clearly defined in the $(k-1)$-TSS model (see Appendix) and an associated probability, $P(b_q/q)$, calculated according to Equation (11). The probability to be associated with those events not seen in the training corpus is then calculated according to Equation (12). Figure 5 shows such a structure for a state q labelled as $\omega_{i-(K-1)}^{i-1}$. The transitions labelled by the $|\Sigma_q|$ words observed

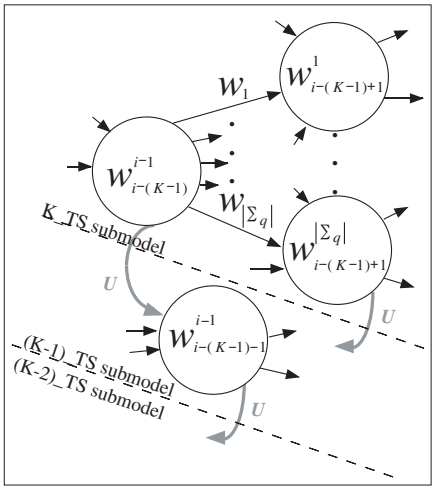

Figure 5. Syntactic back-off smoothing integrated in the automaton structure: transitions labelled by seen events ($\omega_j \in \Sigma_q$) connect each state to states in the same k-TSS submodel, with $k = 1, \ldots, K$. Transitions labelled by unseen events connect to their back-off states in the $(k-1)$-TSS submodel.

at training time after the K-gram labelling q connect it to other states in the same K-TSS submodel. The transition labelled by U connects state q to its back-off state, $\omega_{i-(K-1)+1}^{i-1}$ in the $(K-1)$-TSS submodel.

Figure 6 shows the whole finite network representing the structure of the smoothed and integrated automaton for $K = 4$ when the training set R^+ and the vocabulary Σ of Figure 3 are used. In this figure, each node of the network represents a state of the automaton. Transitions from the state labelled by λ, which represents a void string of words, are represented by links between the root λ and its child nodes. Links connecting node \$ (see Appendix) to its children represent transitions from the initial state. Transitions that correspond to strings of words shorter than K are represented by $|\Sigma_q|$ links connecting each node of the $k = 1, \ldots, K-2$ levels to its $|\Sigma_q|$ children. Nodes of $K-1$ level represent the states associated with word strings of length equal to $K-1$ labelled as $\omega_{i-(k-1)}^{i-1}$. For the sake of clarity, transitions to the final state established for all nodes each time the symbol \$ appears in the training corpus are not drawn in the figure. Instead, nodes representing states with transitions to the node labelled as \$ (see the Appendix) are outlined.

An important advantage of this formulation is that it leads to a very efficient representation of the model parameters learned at training time: probability distributions and model structure. In a first step the structure of a smoothed and integrated K-TSS SFSA is represented by a finite network (see Fig. 6 for an example). This network is derived from an initial trie built from the training set. Then, the initial trie nature of the network allows both the structure of the Smoothed SFSA and the probability distributions to be allocated in a simple linear array. The whole array consists of $|Q^K| \times (|\Sigma_q| + 1)$ positions. Each k-TSS model, $k = 1, \ldots, K$, is represented by a set of states equal to the number of k-grams appearing in the training corpus. A state of the automaton is represented by $|\Sigma_q| + 1$ array rows, each representing one outgoing transition. Each position of the array represents a pair (q, ω) where $q \in Q^K$ and $\omega \in \Sigma_q \cup \{U\}$. It stores the destination state q_d for each $\omega_i \in \Sigma_q \cup \{U\}$ and the value of $P(\omega/q)$ according to Equations (9) and (11). Thus, the smoothed K-TSS SFSA defined in

the Appendix is fully represented and stored in the array. Conventional back-off n-grams do not include any structural parameters. The CMU LM toolbox only associates one probability value and a back-off weight to each n-gram (Clarkson & Rosenfeld, 1997). A complete, detailed procedure for obtaining the representation of the smoothed K-TSS SFSA defined in the Appendix can be found in Torres and Varona (2000) and Varona (2000).

6. Using the smoothed SFSA in a CSR system

As mentioned above, the SFSA defined in Section 3 is deterministic and unambiguous. Thus, each string of words accepted by the automaton labels a unique path from the initial state to a final state. The string $\Omega = la\ de\ la\ vida$, $\Omega \in \Sigma^*$ and $\Omega \in L_{4\text{-TSS}}$, is recognized by the $k = 4$ automaton of Figure 4 through the sequence of state transitions:

$$(\lambda) \xrightarrow{la} (\mathbf{la}) \xrightarrow{de} (\mathbf{lade}) \xrightarrow{la} (\mathbf{ladela}) \xrightarrow{vida} ((\mathbf{delavida})). \tag{14}$$

Thus, at the state labelled as **ladela** there is only one arc available when the word *vida* appears. The probability associated with the string of words $\Omega = la\ de\ la\ vida$ is obtained through Equations (7) and (8). The size of this automaton is $|Q^k| \times |\Sigma_q|\ \forall q \in Q^k$. However, it cannot recognize strings of words, i.e. k-grams, which do not appear in the training sample.

Using Equation (9) instead of (8), this automaton can associate a probability with any string of words in the vocabulary. The smoothed automaton is also a deterministic automaton but now its size is $|Q^K| \times |\Sigma|$, which is prohibitive. The compact representation presented in Section 5 reduces the size of the structure from $|Q^K| \times |\Sigma|$ to $|Q^K| \times (|\Sigma_q| + 1)\ \forall q \in Q^K$. However, the automaton is now non-deterministic. Transitions to back-off states are not labelled by any word of the vocabulary and act as null-transitions. Thus, each string of words accepted by the automaton labels several paths from the initial state to a final state. The string $\Omega = la\ de\ la\ vida$, $\Omega \in \Sigma^*$ and $\Omega \in L_{4\text{-TSS}}$, is now recognized by the smoothed automaton of Figure 6 through several sequences of state transitions. The probability associated with the string of words $\Omega = la\ de\ la\ vida$ should be calculated considering the probabilities associated with all the paths leaving the initial state (\$), reaching a final state and being labelled by this sequence of words, including null-transitions (Fu, 1974). Thus, the non-deterministic automaton is not equivalent to the deterministic one (Llorens, 2000) since the probabilities assigned to the same sequence of words are not the same. As an example, Figure 7 shows a part of the model in Figure 6. Suppose the state labelled by (**ladela**) has been reached. If *vida* is the next word in the sequence then the state labelled as [(**delavida**)] can be reached by two paths: (a) the direct arc labelled by *vida* or (b) the backoff transition to state labelled as (**dela**) followed by the outgoing transition labelled by *vida*. Two more valid paths can be found to reach the state labelled as [(**lavida**)] and two more reaching the state labelled as [(**vida**)] (see Fig. 7).

Another point to be addressed is that the recursive mechanism required by the back-off smoothing technique [see Equation (9)] is not well integrated in the compact representation. In the example of Figure 7, the path that gives the probability corresponding to the smoothing technique [Equation (9)] is [(**ladela**)] $\xrightarrow{vida}$ [(**delavida**)] since *vida* is a *seen* event at state labelled by (**ladela**). However, all possible paths are considered to associate a probability with a word sequence through a non-deterministic automaton (Fu, 1974). Thus, the *if ... then ... else* structure required by any smoothing technique cannot be implemented in a non-deterministic background (Llorens, 2000) due to the multiple path choice for the same string of words.

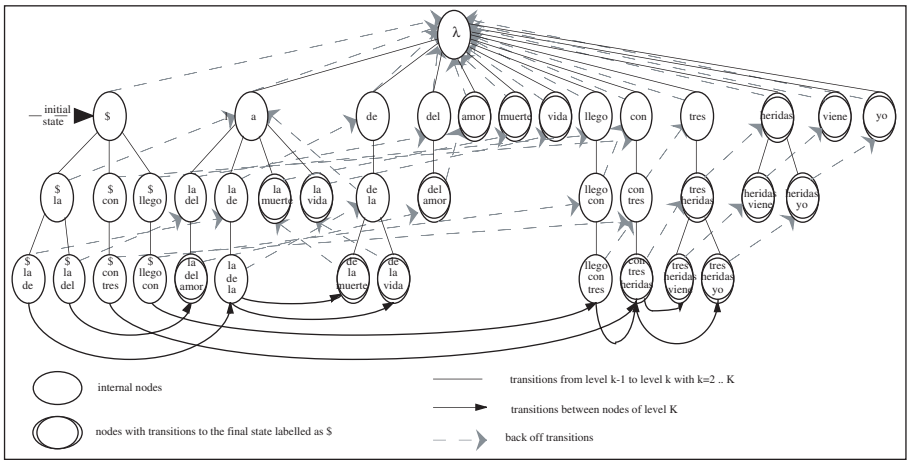

Figure 6. The finite network representing the structure of the smoothed and integrated automaton ($K = 4$) obtained from the training set R^+ of Figure 3.

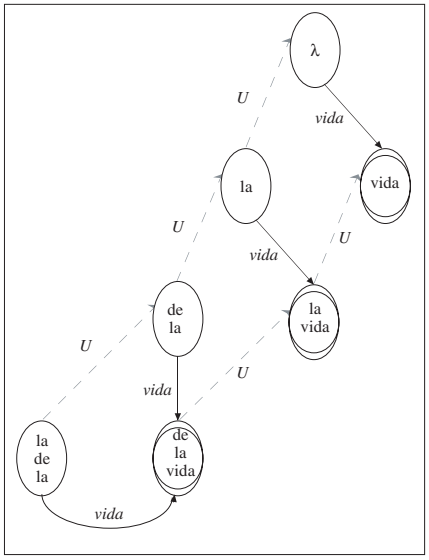

Figure 7. Part of the model in Figure 6. Several paths represent the string of words $\Omega =$ *la de la vida*.

The Viterbi decoding algorithm finds the most likely path through a trellis. Thus, only the best sequence of states is considered when obtaining $\hat{\Omega}$ through Equation (2). If the LM is represented by a non-deterministic SFSA, then the decoding algorithm is taking the final decision about valid or non valid sequences of states, and consequently sequences of words, by considering only the most likely state sequences. This procedure does not guarantee the same probability distribution as the deterministic model (Riccardi *et al.*, 1996; Llorens, 2000). The path leading to the correct application of the smoothing technique is not guaranteed to be chosen in this case (Riccardi *et al.*, 1996; Llorens, 2000). Some proposals to address this problem chose compact representations, i.e. non-deterministic automata, due to the major

```
Function δ(q ∈ Q^K;ω ∈ Σ):(d ∈ Q^K; P ∈ [0...1]);
var q_aux ∈ Q^K;P_aux ∈ [0...1];
begin
if ω ∈ Σ_q   then    δ_P ← array_prob[q, w];          (*seen events*)
                     δ_d ← array_dest[q, w]
             else    P_aux ← array_prob[q, U];        (*unseen events*)
                     q_aux ← array_dest[q, U]
                     while ω ∉ Σ_q_aux do
                           P_aux ← P_aux × array_prob[q_aux, U];
                           q_aux ← array_dest[q_aux, U];
                     end_while
                     δ_P ← P_aux × array_prob[q_aux, ω];
                     δ_d ← array_dest[q_aux, ω];
end_if
end_δ.
```

Figure 8. A simple search function to compute Equation (15).

savings in memory. But they also use the non-deterministic network at decoding time. In such cases, only approximate *n*-gram probability distributions are managed (Placeway *et al.*, 1993; Riccardi *et al.*, 1996; Bonafonte & Mariño, 1998; Suzuki & Aso, 1999; Llorens, 2000).

One way to deal with these problems is to develop a specific parsing method which implements the smoothing technique in an exact way while keeping the compact representation of the automaton. This procedure should guarantee that the probability to be assigned to a given *k*-gram is the same as the probability assigned by the full deterministic automaton. This goal is achieved by a simple search function through the array representing the non-deterministic and smoothed SFSA in Figure 5 (Torres & Varona, 2000; Varona, 2000).

To complete the model representation the transition function δ of the smoothed and integrated K-TSS automaton (see the Appendix) should then be established and represented for each state $q \in Q^K$ and for each $\omega \in \Sigma'$. This transition function defines a destination state q_d and a probability $P(\omega/q)$ associated with each pair (q, ω) $\forall q \in Q^K$ and $\forall \omega \in \Sigma'$:

$$\delta(q, \omega) = (q_d, P(\omega/q)) \qquad \forall q \in Q^K \wedge \forall \omega \in \Sigma' \qquad q_d \in Q^K. \tag{15}$$

When seen events appear the destination state q_d corresponds to a state in the same k-TSS submodel (see Fig. 5). Thus, it can be directly found as the destination index of the array position (ω, q) (Torres & Varona, 2000). In the same way the value of $P(\omega/q)$, computed according to Equation (9) for $\omega \in \Sigma_q$, can be directly found as the probability value at the array position (ω, q) (Torres & Varona, 2000). However, when unseen events appear q_d should be found in the $(k-1)$-TSS submodel and thus neither q_d nor $P(\omega/q)$ values are directly found in the array. A simple search function through the array is then required. This function, represented in Figure 8, searches backwards across the back-off states, i.e. transitions through the U symbol, until the word ω is found as a seen event for a state q in a lower level ($k < K$), i.e. $\omega \in \Sigma_q$ (see Fig. 5). State q will be then the destination state q_d searched for. The $P(\omega/q)$ value should be computed according to Equation (12) in this case. Thus, Equation (12) is recursively calculated while q_d is found by the search function. This procedure, represented by Equation (15), is described by function δ in Figure 8: δ_d stands for the destination state q_d and δ_P for the probability $P(\omega/q)$.

When the next word to be processed is known, as in text processing, the function repre-

sented in Figure 8 is the best way to compute the smoothed probability while keeping the compact representation. In speech recognition applications the word to be processed is unknown and as a consequence the structure needs to be dynamically expanded at decoding time (Varona, 2000). The search function provides in this case an array representing different sequences of decoded words obtained by Equation (15). However, this search function is needed for active paths only when a simple beam search strategy (Ney, 1992; Steinbiss, Tran & Ney, 1994) is used to reduce the computational cost. As a consequence, the size of the real search space does not increase, even when high values of K are used. The experimental evaluation carried out (see Section 7) shows this behaviour.

7. Experimental evaluation

An experimental evaluation of k-TSS languages models was carried out on two Spanish databases. The first one was first presented in English (Feldman, Lakoff, Stolcke & Weber, 1990) and consists of a set of simple Spanish sentences describing visual scenes (miniature language acquisition task—MLA). The training set consisted of 9150 sentences that were randomly generated by using a context-free model of the language. It includes 147 002 words and a very limited vocabulary size (29 words). The second corpus (BDGEO) is a task-oriented Spanish speech corpus (Díaz *et al.*, 1993) with a medium size vocabulary of 1212 words. This corpus represents a set of queries to a Spanish geography database. This is a specific task designed to test integrated systems (acoustic, syntactic and semantic modelling) in automatic speech understanding. The training corpora also consisted of 9150 sentences including 82 000 words in this case.

Tables I and II show the number of states, $|Q^K|$, for several K-TSS integrated models, with $K = 1 \ldots 6$ for MLA and BDGEO tasks respectively. The increase of $|Q^K|$ with K depends on the corpus size and corpus structure. However, the rate between the number of seen events and the number of possible events decreases with K in any case. Consequently, the increase of $|Q^K|$ with K remains quite limited. The smoothed K-TSS SFSA formulation presented in Section 5 along with the proposed representation (see Fig. 5) reduces the number of parameters to be handled from $|Q^K| \times |\Sigma|$ to $|Q^K| \times (|\Sigma_q| + 1)$. Tables I and II show that this reduction leads to memory requirements that can be easily managed even when high values of K are considered. These tables also show the test set perplexity when Equation (9) was used to estimate the probability distributions for several K-TSS models. A set of 500 new sentences including 8397 words was used to evaluate the test set perplexity of the MLA corpus and a set of 1193 new sentences and 13 687 words was used to evaluate the test set perplexity of the BDGEO corpus. Perplexity values decrease as K increases, even for high values of K, for MLA corpus (see Table I). This behaviour is due to the simplicity of the syntactic structure of these sentences, especially for high values of K. The perplexity values are stable for $K > 3$ for BDGEO corpus (see Table II).

These experiments were also carried out using standard back-off n-gram models along with the Witten–Bell discounting. Tables I and II show the test set perplexity obtained through the CMU toolkit (Clarkson & Rosenfeld, 1997) for several n-gram models, with $n = 1, \ldots, 6$ for MLA and BDGEO tasks. K-TSS and n-gram models achieved similar results, as expected. Tables I and II also show that the smoothed and integrated K-TSS model needs only a little more memory than a conventional back-off n-gram using the CMU toolkit (Clarkson & Rosenfeld, 1997). In fact, conventional back-off n-grams do not include any structural parameters. Only a probability value and a back-off weight is associated with each n-gram in the CMU toolkit. However, the structure of K-TSS SFSA defined in the Appendix is fully

TABLE I. Perplexity evaluation and sizes of several K-TSS, with $K = 1 \ldots 6$ (see Fig. 5) and CMU n-grams models (Clarkson & Rosenfeld, 1997) for MLA task. For comparison purposes, the size of a theoretic and deterministic full network is also provided

K	$\lvert Q^K \rvert$	Full network	K-TSS			n-grams (CMU)	
		$\lvert Q^K \rvert \times \lvert \Sigma \rvert$	$\lvert Q^K \rvert \times (\lvert \Sigma_q \rvert + 1)$	Memory (Kb)	PP	Memory (Kb)	PP
2	31	899	214	3.0	3.25	2.04	4.03
3	173	8590	850	11.9	3.08	8.33	3.12
4	643	18647	2542	35.6	2.76	25.5	2.90
5	1808	52432	6642	93.0	2.62	69.75	2.77
6	4518	131022	15614	218.6	2.56	176.68	2.61

TABLE II. Perplexity evaluation and sizes of several K-TSS, with $K = 1 \ldots 6$ (see Fig. 5) and CMU n-grams models (Clarkson & Rosenfeld, 1997) for BDGEO task. For comparison purposes, the size of a theoretic and deterministic full network is also provided

K	$\lvert Q^K \rvert$	Full network	K-TSS			n-grams (CMU)	
		$\lvert Q^K \rvert \times \lvert \Sigma \rvert$	$\lvert Q^K \rvert \times (\lvert \Sigma_q \rvert + 1)$	Memory (Mb)	PP	Memory (Mb)	PP
2	1213	$14.60\ 10^5$	9285	0.13	13.10	0.09	13.03
3	7479	$90.34\ 10^5$	30714	0.43	7.53	0.32	7.54
4	21551	$26.03\ 10^6$	67857	0.95	6.95	0.80	7.17
5	42849	$51.76\ 10^6$	120714	1.69	6.90	1.52	7.22
6	69616	$80.09\ 10^6$	182142	2.55	6.90	2.61	7.37

represented and stored. A destination state q_d and the value of $P(\omega/q)$ [see (9) and (11)] are stored for each $\omega_i \in \Sigma_q \cup \{U\}$ (Torres & Varona, 2000). This leads to a more effective network search at decoding time. It also allows a straightforward integration of other knowledge sources.

In a second series of experiments, the K-TSS SFSA were integrated into a CSR system. Each transition of the automaton was replaced by a chain of hidden Markov models representing the acoustic model of each phonetic unit of the word. Then the decoding scheme represented in Equation (2) was approximated by using the time-synchronous Viterbi algorithm. In such a scheme, the transition through each word of the vocabulary should be evaluated each time the system considers an LM state transition $\delta(q/\omega)$. Thus, the search function presented in Figure 8 was used to obtain for state q the following state q_d and the associated probability $P(\omega/q)$ for all the words in the vocabulary.

Two new test sets were used in this case: the first consisted of 1600 sentences from the MLA task, uttered by 16 speakers, and the second of 600 sentences from the BDGEO task, uttered by 12 speakers. In both cases the acoustic models were previously trained over 1529 sentences, phonetically balanced and uttered by 47 different speakers, involving around 60 000 phones.

A Silicon Graphics O2 machine with an R10000 processor was used in these experiments. In order to reduce the computational cost a beam-search algorithm was applied with three different widths: a narrow beam factor (*bf*) of 0.6, an intermediate of 0.5 and a wide one of 0.4. The beam-search algorithm eliminates the less probable paths of the trellis. The word error rate (WER) obtained when several K-TSS languages were integrated into the described CSR system is shown in Table III (MLA task) and Table IV (BDGEO task). These tables also show the average time required to decode a frame (in mseconds) and the average number of active nodes in the lattice, including acoustic and language model states.

Tables III and IV show that the use of the proposed smoothed SFSA representation pro-

TABLE III. Experimental results when several K-TSS models were integrated into a CSR system: MLA task

K	Average number of active nodes			Average time per frame (msec)			WER		
	$bf = 0.6$	$bf = 0.5$	$bf = 0.4$	$bf = 0.6$	$bf = 0.5$	$bf = 0.4$	$bf = 0.6$	$bf = 0.5$	$bf = 0.4$
2	21	39	64	1	3	4	7.53	5.01	4.60
3	26	47	80	2	3	5	7.48	4.83	4.36
4	28	50	87	2	4	6	7.17	4.33	3.62
5	29	52	92	2	4	7	6.76	3.64	3.26
6	30	54	98	3	5	7	6.82	3.51	2.68

TABLE IV. Experimental results when several K-TSS models were integrated into a CSR system: BDGEO task

K	Average number of active nodes			Average time per frame (msec)			WER		
	$bf = 0.6$	$bf = 0.5$	$bf = 0.4$	$bf = 0.6$	$bf = 0.5$	$bf = 0.4$	$bf = 0.6$	$bf = 0.5$	$bf = 0.4$
2	114	218.21	526.28	6.6	11.8	20.54	25.1	15.95	14.29
3	90	179.01	467.66	6.0	10.9	17.51	25.5	10.85	9.45
4	89	177.99	469.60	6.2	11.2	16.71	26.3	10.12	8.58
5	90	180.57	478.33	6.5	11.6	16.73	27.0	10.25	8.72
6	91	182.32	490.50	6.8	12.0	16.51	27.1	10.66	9.07

vides additional time reductions since the search function through the array (Fig. 8) is only needed for active paths. Thus, smoothed K-TSS models with high values of K need more memory (see Tables I and II) but they can be easily integrated into a CSR system with no significant increase in the decoding time, since the average number of active nodes does not increase as $|Q^K|$ does. This is a very important result since it demonstrates that a compact representation of the non-deterministic model (see Fig. 5) can be dynamically expanded at decoding time at a feasible computational cost. The use of the function in Figure 8 along with a simple beam-search algorithm results in a relatively constant size of the search space whereas the performance of the system increases with k. In this way, the Viterbi decoding algorithm can be used to decode a compact and non-deterministic SFSA which obtains the same probability distributions as the deterministic SFSA. Moreover, the probability assigned by the smoothing technique is guaranteed to be considered at each step of the LM decoding procedure.

These experiments also show that both the system performance and the average number of active nodes increase with K for the MLA task (Table III). The lowest word error rates were achieved for $K = 6$. However, the system performance behaviour is not the same for the BDGEO task. In this case, both the WER and the average number of active node values decreased as K increased up to a certain value of K(see Table IV); subsequently, both values remain practically constant. The lowest word error rates were achieved for $K = 4$ in this task.

8. Concluding remarks

The aim of a CSR system is to find the best interpretation of the input speech data in terms of knowledge sources such as language model, pronunciation lexicon and inventory of subword units. This objective is represented by the well-known Bayes' formula, which represents an integrated model involving global decisions that explicitly take account of the constraints given by the knowledge sources. It is well known that a straightforward way of dealing with such an integration is to use SFSA. On the other hand, language constraints are better mod-

elled under a syntactic approach. The aim of this work is to show the ability of stochastic regular grammars to generate accurate LMs which can be well integrated into this scheme. For this purpose, a syntactic version of the well-known n-gram models, k-TSS language models, has been used. The complete definition of a k-TSS SFSA is provided in the paper. A syntactic back-off smoothing technique has been applied to the SFSA, modifying the probability distribution to consider unseen events. Finally, a self-contained smoothed model integrating K k-TSS automata, where $k = 1, \ldots, K$, has been also defined. This formulation leads to a very compact representation of the model parameters learned at training time: probability distribution and model structure.

Another difficulty arising in representing a language model through a stochastic finite state network is that the recursive schema usually involved in the smoothing procedure must be adopted in the finite state formalism to achieve an efficient implementation of the backing-off mechanism. The dynamic expansion of the structure at decoding time allows the Viterbi algorithm to correctly manage the *if. . .then. . . else* schema of the back-off procedure. Thus, a one-step decoding algorithm can be used to decode a compact and non-deterministic SFSA, obtaining the same probability distributions as the deterministic one.

An experimental evaluation of the proposed formulation has been carried out on two Spanish corpora. These experiments show that the dynamic expansion of the structure at decoding time do not lead to an increase in the size of the search space. The number of active nodes in the lattice did not increase as $|Q^k|$ did, whereas the system performance increased with k. The system performance shown in the paper with high values of k is consistent enough for good system behaviour with large vocabulary tasks to be expected, even though experiments of this kind may also be required. In fact, from a computational point of view, increasing the value of k could be considered as equivalent to increasing the size of the vocabulary of the task.

This work shows that regular grammars can generate accurate language models, k-TSS, that can be efficiently represented and managed in real speech recognition systems, even for high values of k, leading to very good system performances. Thus, efficient integration of LM generated by more powerful regular grammars can be attempted in the same way. Future works also include straightforward integration of other knowledge sources.

The authors would like to thanks Dr F. Casacuberta and Dr E. Vidal for their comments and suggestions during the composition of this paper. They would also like to thank the reviewers and the editor for their suggestions on improving the quality of the paper. This work has been partially supported by the Spanish CICYT under grant TIC98-0423-C06-03.

References

Aubert, X. (2000). A brief overview of decoding techniques for large vocabulary continuous speech recognition. *Automatic Speech Recognition: Challenges for the New Millenium, ISCA ITRW ASR2000 Workshop*, Paris, pp. 91–96.

Bahl, L. R., de Genaro, S. V., Gopalakrishnan, P. S. & Mercer, L. R. (1993). A fast approximate acoustic match for large vocabulary speech recognition. *IEEE Transactions on Speech and Audio Processing*, **I**, 59–67.

Bonafonte, A. & Mariño, J. B. (1998). Using x-grams for efficient speech recognition. *Proceedings of International Conference on Speech and Language Processing*.

Bordel, G. (1996). *Aprendizaje automático de modelos K-explorables estocásticos en reconocimiento del habla*. PhD Thesis, Universidad del País Vasco.

Bordel, G., Torres, I. & Vidal, E. (1994). Back-off smoothing in a syntactic approach to language modelling. *Proceedings of International Conference on Speech and Language Processing*, pp. 851–854.

Brown, P. F., deSouza, P. V., Mercer, R. L., dellaPietra, V. J. & Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, **18**, 467–479.

Caseiro, D. & Trancoso, I. (2000). A decoder for finite-state structured search spaces. *Automatic Speech Recognition: Challenges for the New Millenium, ISCA ITRW ASR2000 Workshop*, Paris, pp. 35–39.

Chandhuri, R. & Booth, T. L. (1986). Approximating grammar probabilities: solution of a conjecture. *Journal of the ACM*, **33**, 702–705.

Channon, C. & Weaver, W. (1981). *Teoría matemática de la comunicación*, Ediciones Forja, S.A., Madrid.

Clarkson, P. & Rosenfeld, R. (1997). Statistical language modeling using the CMU-Cambridge toolkit. *Proceedings of European Conference on Speech Technology*, pp. 2707–2710.

Díaz, J. E., Rubio, A., Peinado, A., Segarra, E., Prieto, N. & Casacuberta, F. (1993). Development of task oriented spanish speech corpora. *Proceedings of European Conference on Speech Technology*.

Dupont, P. & Miclet, L. Inférence grammaticale régulière. fondements théoriques et principaux algorithmes. Rapport de reserche N 3449, Institut National de recherche en informatique et en automatique, Rennes.

Feldman, J. A., Lakoff, G., Stolcke, A. & Weber, S. H. (1990). Miniature language acquisition: a touch-stone cognitive science. Technical Report TR-90-009, ICSI, Berkeley, CA.

Forney, G. D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, **61**, 268–278.

Fu, K. S. (1974). *Syntactic Methods in Pattern Recognition, Mathematics in Science and Engineering*. Academic Press, New York.

García, P. & Vidal, E. (1990). Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, 920–925.

Hopcroft, J. E. & Ullman, J. D. (1979). *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley.

Jelinek, F. (1976). Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, **64**, 532–556.

Jelinek, F. (1985). Markov source modelling of text generation. In *The Impact of Processing Techniques on Communication*, (Skwirzynski, J. K. and Nijhoff, M., eds), pp. 569–598. Dordrecht, The Netherlands.

Jelinek, F. (1999). *Statistical Methods for Speech Recognition, Language, Speech and Communication Series*, 2nd edition. The MIT Press, Cambridge, MA.

Jelinek, F., Lafferty, J. D. & Mercer, R. L. (1992). Basic methods of probabilistic context free grammars. In *Speech Recognition and Understanding*, Springer-Verlag, The Netherlands.

Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recogniser. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-**35**, 400–401.

Llorens, D. (2000). *Suavizado de autómatas y traductores finitos estocásticos*. PhD Thesis, Universidad Politécnica de Valencia.

Mohri, M., Pereira, F. & Riley, M. (2000). Weighted finite-state transducers in speech recognition. In *Automatic Speech Recognition: Challenges for the New Millenium, ISCA ITRW ASR2000 Workshop*, Paris, pp. 97–106.

Ney, H. (1992). Stochastic grammars and pattern recognition. In *Speech Recognition and Understanding. Recent advances, NATO-ASI Series*. (Laface, P. and de Mori, R., eds), pp. 319–344. Springer Verlag.

Ney, H., Martin, S. & Wessel, F. (1997). Statistical language modeling using leaving-one-out. In *Corpus-Based Methods in Language and Speech Processing*, (Young, S. and Bloothooft, G., eds), pp. 174–207. Kluwer Academic Publishers.

Niesler, T. R. & Woodland, P. C. (1999). Variable-length category *n*-gram language models. *Computer Speech and Language*, **13**, 99–124.

Pereira, Fernando C. N. & Riley, Michael D. (1997). Speech recognition by composition of weighted finite automata. In *Finite-State Language Processing, Language, Speech and Communication Series*. (Roche, Emmanuel and Schabes, Yves, eds), pp. 431–453. The MIT Press, Cambridge, MA.

Placeway, P., Schwartz, R., Fung, P. & Nguyen, L. (1993). The estimation of powerful language models from small and large corpora. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pp. 33–36.

Prieto, N. & Vidal, E. (1992). Learning languages models trough the ecgi method. *Speech Communication*, **11**, 299–309.

Rabiner, L. & Juang, B. H. (1993). *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, NJ.

Riccardi, G., Bochieri, E. & Pieraccini, R. (1995). Non deterministic stochastic language models for speech recognition. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pp. 237–240.

Riccardi, G., Pieraccini, R. & Bocchieri, E. (1996). Stochastic automata for language modeling. *Computer Speech and Language*, **10**, 265–293.

Schwartz, R. & Austin, S. (1991). A comparison of several approximation algorithms for finding multiple n-best sentence hypotheses. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 701–704.

Segarra, E. (1993). *Una aproximación inductiva a la comprensión del discurso continuo*. PhD Thesis, Universidad Politécnica de Valencia.

Steinbiss, V., Tran, B. & Ney, H. (1994). Inprovement in beam search. *Proceedings of International Conference on Speech and Language Processing*, pp. 2143–2146.

Suzuki, M. & Aso, H. (1999). An automatic acquisition method of statistic finite-state automaton for sentences. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*.

Torres, I. & Varona, A. (2000). An efficient representation of k-TSS language models. *Computación y Sistemas (Revista Iberoamericana de Computación)*, **3**, 237–244.

Varona, A. (2000). *Modelos k-explorables en sentido estricto integrados en un sistema de reconocimiento automático del habla*, Servicio editorial de la Universidad del Pas Vasco, Leioa, Spain.

Varona, A. & Torres, I. (1999). Using smoothed k-TLSS language models in continous speech recognition. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pp. 729–732.

Varona, A. & Torres, I. (2000*a*). Delimited smoothing technique over pruned and not pruned syntactic language models: perplexity and wer. In *Automatic Speech Recognition: Challenges for the New Millenium, ISCA ITRW ASR2000 Workshop*, Paris, pp. 69–76.

Varona, A. & Torres, I. (2000*b*). Evaluating pruned k-TSS language models: preplexity and word recognition rates. In *Pattern Recognition and Applicactions, Frontiers in Artificial Intelligence and Applications Series*, pp. 193–202. IOS Press, The Netherlands.

Vidal, E., Casacuberta, F. & García, P. (1995). Syntactic learning techniques for language modelling and acoustic-phonetic decoding. In *Speech Recognition and Coding. New Advances and Trends, Series F: Computer and Systems Sciences*. (Rubio, A. and López, J. M., eds), Springer-Verlag.

Zhao, R., Kenny, P., Labute, P. & O'Shoughnessy, D. (1993). Issues in large scale statistical language modeling. *Proceedings of European Conference on Speech Technology*, pp. 965–968.

(Received 23 November 1999 and accepted for publication 8 February 2001)

Appendix: The smoothed and integrated model

A smoothed K-TSS SFSA is a self-contained model that integrates K k-TSS automata, where $k = 1, \ldots, K$ in a unique automaton. It is defined by a five-tuple $(\Sigma', Q^K, q_0, q_f, \delta^K)$ where:

- $\Sigma' = \Sigma \cup \{\$\}$ being $\Sigma = \{\omega_j\}$, $j = 1, \ldots, |\Sigma|$, is the vocabulary, that is the set of words appearing in the training corpus. Null symbol \$ has been included in the training corpus to isolate each sentence from its neighbors. This symbol can be considered as the first and last word of each sentence.
- Q^K is the state set of the automaton. Each state represents a string of words $\omega_{i-k}\omega_{i-(k-1)}, \ldots, \omega_{i-1}$, $k = 1, \ldots, K-1$, with a maximum length of $K-1$, where i stands for a generic index in any string $\omega_1 \ldots \omega_i \ldots$ appearing in the training corpus. Such a state is labelled as ω_{i-k}^{i-1}. States representing the initial strings of training sentences are labelled as $\$\omega_{i-k}^{i-1}$ where $k = 1, \ldots, K-2$ to guarantee a maximum length of $K-1$. A special state labelled as λ represents a void string of words.
- The automaton has a unique initial and final state $q_0 \equiv q_f \in Q^K$, which is labelled as \$. This allows us consecutively to parse sets of sentences, while discarding contextual information between them. The initial state is different from the state labelled as λ because the probability $P(\omega_i/\lambda)$ is the estimated probability $P(\omega_i)$, whereas $P(\omega_i/\$)$ is the probability associated with ω_i when it is the initial word in a sentence. The only exception is the 1-TSS model with a unique state λ, which is the initial and final state.
- δ^K is the transition function $\delta^K : Q^K \times (\Sigma' \cup \{U\}) \rightarrow Q^K \times [0 \ldots 1]$. $\delta^K(q, \omega_i) = (q_d, P(\omega_i/q))$ defines a destination state $q_d \in Q^K$ and a probability $P(\omega_i/q) \in [0 \ldots 1]$ to be assigned to each element $(q, \omega_i) \in Q^K \times (\Sigma \cup \{\$U\})$. Each transition represents a k-gram, $k = 1, \ldots, K$; it is labelled by its last word ω_i and connects two states labelled with up to $K-1$ words. Several kinds of transition can be found in the model:

(a) Transitions from the special state labelled as λ, which represents a void string of words, to the $|\Sigma|$ states labelled by each word of the vocabulary $\omega_j \in \Sigma$, $j = 1, \ldots, |\Sigma|$. The probability associated with each transition represents the frequency of the corresponding word in the training corpus:

$$\delta^K(\lambda, \omega_j) = (\omega_j, P(\omega_j/\lambda)) = (\omega_j, P(\omega_j)) \qquad j = 1, \ldots, |\Sigma|. \tag{16}$$

(b) Transitions from the initial state labelled as \$ that correspond to the set of words appearing in initial position of a sentence:

$$\delta^K(\$, \omega_i) = (\$\omega_i, P(\omega_i/\$)). \tag{17}$$

(c) Transitions that correspond to strings of words shorter than K connecting states associated with string lengths up to $K-2$:

$$\begin{cases} \delta^K(\omega_{i-k}^{i-1}, \omega_i) = (\omega_{i-k}^{i}, P(\omega_i/\omega_{i-k}^{i-1})) & k = 1, \ldots, K-2 \\ \delta^K(\$\omega_{i-k}^{i-1}, \omega_i) = (\$\omega_{i-k}^{i}, P(\omega_i/\$\omega_{i-k}^{i-1})) & k = 1, \ldots, K-3. \end{cases} \tag{18}$$

These transitions come from the $K-1$ automata corresponding to k-TSS models with $k < K$.

(d) Regular transitions that correspond to strings of words of length K connecting states associated with string lengths equal to $K-1$:

$$\begin{cases} \delta^K(\omega_{i-(K-1)}^{i-1}, \omega_i) = (\omega_{i-(K-1)+1}^{i}, P(\omega_i/\omega_{i-(K-1)}^{i-1})) \\ \delta^K(\$\omega_{i-(K-2)}^{i-1}, \omega_i) = (\$\omega_{i-(K-2)}^{i}, P(\omega_i/\$\omega_{i-(K-2)}^{i-1})). \end{cases} \tag{19}$$

(e) Transitions to the final state $q_f (\equiv q_0)$ labelled by \$:

$$\begin{cases} \delta^K(\omega_{i-k}^{i-1}, \$) = (q_f, P(\$/\omega_{i-k}^{i-1})) & k = 1, \ldots, K-1 \\ \delta^K(\$\omega_{i-k}^{i-1}, \$) = (q_f, P(\$/\$\omega_{i-k}^{i-1})) & k = 1, \ldots, K-2. \end{cases} \tag{20}$$

(f) Transitions to the back-off state: under the syntactic back-off smoothing procedure (see Section 4), the probability $P(\omega_i/q)$ to be associated with a transition $\delta^K(q, \omega_i)$ is estimated according to Equation (9). Each state of the automaton $q \in Q^K$, except the state labelled by λ, should add a new transition to its back-off state b_q:

$$\delta^K(q, U) = (b_q, P(b_q/q)) \tag{21}$$

where U represents any unseen event associated with state q which is labelled by a word ω_j in $(\Sigma' - \Sigma_q)$. The back-off state b_q associated with each state q can be found in the $(k-1)$-TSS model. Thus, for states associated with string lengths up to $K-2$:

$$\begin{aligned} q \equiv \omega_{i-k}^{i-1} &\Rightarrow b_q \equiv \omega_{i-k+1}^{i-1} \quad k = 1, \ldots, K-2 \\ q \equiv \$\omega_{i-k}^{i-1} &\Rightarrow b_q \equiv \$\omega_{i-k+1}^{i-1} \quad k = 1, \ldots, K-3 \end{aligned} \tag{22}$$

and for ordinary states associated with string lengths equal to $K-1$:

$$\begin{aligned} q \equiv \omega_{i-(K-1)}^{i-1} &\Rightarrow b_q \equiv \omega_{i-(K-1)+1}^{i-1} \\ q \equiv \$\omega_{i-(K-2)}^{i-1} &\Rightarrow b_q \equiv \$\omega_{i-(K-2)+1}^{i-1}. \end{aligned} \tag{23}$$

Finally, the back-off state associated with the initial, and final, state labelled as \$ is the special state labelled as λ. This state has no associated back-off state since it represents the void string.