Non Deterministic Stochastic Language Models for Speech Recognition

G. Riccardi, E. Bocchieri, R. Pieraccini
Speech Research Department
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974, USA

1. ABSTRACT

Traditional stochastic language models for speech recognition (i.e. n-grams) are deterministic, in the sense that there is one and only one derivation for each given sentence. Moreover a fixed temporal window is always assumed in the estimation of the traditional stochastic language models. This paper shows how non-determinism is introduced to effectively approximate a back-off n-gram language model through a finite state network formalism. It also shows that a new flexible and powerful network formalization can be obtained by releasing the assumption of a fixed history size. As a result, a class of automata for language modeling (Variable Ngram Stochastic Automata) is obtained, for which we propose some methods for the estimation of the transition probabilities . VNSAs have been used in a spontaneous speech recognizer for the ATIS task. The accuracy on a standard test set is presented in this paper.

2. INTRODUCTION

The n-gram language model is based on the assumption that the probability of a word in a sentence $(w_1, \ldots, w_{N-1}, w_N)$ depends only on the $n-1$ previous words. Under this assumption, it can be shown that:

$$P(w_1, \ldots, w_{N-1}, w_N) = \prod_{i=1}^{i=N} P(w_i | w_{i-n+1}, \ldots, w_{i-1}) \quad (1)$$

The maximum likelihood estimation of the conditioned probabilities on the right hand side of (1) can be easily obtained by counting the occurrences of word sequences (n-tuples) in the training data:

$$\hat{P}(w_i | w_{i-n+1}, \ldots, w_{i-1}) = \frac{count(w_{i-n+1}, \ldots, w_{i-1}, w_i)}{count(w_{i-n+2}, \ldots, w_{i-1}, w_i)} \quad (2)$$

The main problem in the estimation (2) is that not all the possible n-tuples, $(w_{i-n+1}, \ldots, w_{i-1}, w_i)$ have been observed in the training data (*unseen events*). Since a language model must provide a non zero probability for any possible sentence, a technique has to be used in order to give a non zero probability to the unseen n-tuples. This is generally obtained through a *backoff* approach, namely using the probability estimates of lower order m-tuples ($m < n$) when the original n-tuple was not observed in the training corpus. Thus in the n-gram language models, the backoff approach leads to estimation formulae of the kind:

if $\hat{P}(w_i | w_{i-n+1}, \ldots, w_{i-1}) \neq 0$
$\quad P(w_i | w_{i-n+1}, \ldots, w_{i-1}) = \hat{P}(w_i | w_{i-n+1}, \ldots, w_{i-1})$
else if $\hat{P}(w_i | w_{i-n+2}, \ldots, w_{i-1}) \neq 0$
$\quad P(w_i | w_{i-n+1}, \ldots, w_{i-1}) = \alpha_1 \hat{P}(w_i | w_{i-n+2}, \ldots, w_{i-1})$
...
else

$$P(w_i | w_{i-n+1}, \ldots, w_{i-1}) = \prod_{i=1}^{i=n-1} \alpha_i \hat{P}(w_i) \quad (3)$$

Where α_i are suitable parameters that satisfy:

$$\sum_{w \in V} P(w_i = w | w_{i-n+1}, \ldots, w_{i-1}) = 1. \quad (4)$$

This work develops a unified representation of general backoff n-gram language models through a non deterministic stochastic finite state network. The purpose is twofold:

1. To make the decoder (recognizer) algorithm independent of the order of the language model used.
2. To improve the efficiency of the decoder search algorithm. In general, speech recognition is performed by searching for the word sequence that maximizes a MAP criterion. With our approach, this process is formalized as the search of the "best" path in a network. Therefore, we can increase the decoder efficiency by proper selection of the finite state network structure.

The difficulties arising in representing a set of backoff probabilities through a stochastic non deterministic finite-state network are the following:

0-7803-2431-5/95 $4.00

1. The straightforward full network representation, where each probability $P(w_i|w_{i-n+1},\ldots,w_{i-1})$ is associated to an arc, is not possible, because it requires a number of arcs proportional to the n-th power of the vocabulary size.

2. The conditions on the left side of equation 3 must be taken into account in the finite state formalism.

In this paper we show how a stochastic non-deterministic automaton is used to effectively approximate a n-gram language model. We also show that this automaton can represent a whole class of language models where the assumption of a fixed history size is released. This automaton will be called Variable Ngram Stochastic Automaton.

3. VNSA FOR BIGRAM LANGUAGE MODELS

Backoff strategies can be implemented in an n-gram network in such a way that the speech recognizer does not have to handle the estimation formulas 3 directly, but is has only to perform the search through a finite state stochastic network. In particular we want to eliminate conditions (e.g if$\hat{P}(w_i|w_{i-n+1},\ldots,w_{i-1}) \neq 0$) like those in equation 3. Assume we have a bigram network like the one shown in Fig. 1. The vocabulary is composed of three words (a, b, and c) and only the events ab, ac, bc, cb were observed with associated probability $P(b|a)$, $P(c|a)$, $P(b|c)$ and $P(c|b)$. Using this network for parsing (recognizing) a sequence of words will fail when the sequence contains bigrams that were not observed, like in the sequence $aabacb$. A solution is to add *all* the missing transitions and to

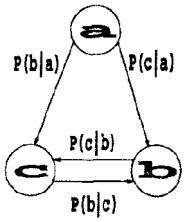

Figure 1: Example of a bi-gram network

compute their probability according to the backoff estimation technique. This solution (*full n-gram network*) cannot be realized in practice since the total number of transitions (i.e. N_V^n, where N is the size of vocabulary V) is too big for the implementation of the network in a real application. With a vocabulary of 1,000 words, a tri-gram language model will occupy at least 4 Gbytes of memory. The backoff probabilities defined by equation 2 can instead be approximated with the network shown in Fig. 2. It should be noticed that there is

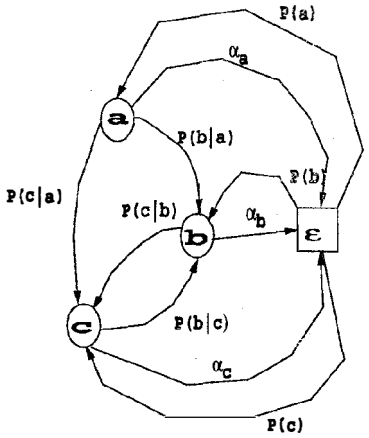

Figure 2: Example of a bi-gram network that approximates a full network

only a transition out of each node (arcs with associated probability P_a, P_b) and P_c) to represent all the missing transitions. All those transitions are leading to a common *null* state (i.e. a state that does not recognize any word) that is represented by the symbol ϵ. The null node is then connected to each state in the network (arcs with associated probability γ_a, γ_b and γ_c). This network corresponds to a second order VNSA.

Non determinism of VNSAs

It has to be noticed that a VNSA is an aproximation of the probability discounting schema of equation 3. In fact the resulting automaton is non deterministic, meaning that for a given input sentence there are several possible sequences of states that recognize it. For instance, in the previous example, the sentence **abbc** can be recognized through the sequences of states **abϵbc**, **ϵabϵbc**, **aϵbϵbc**, **abϵbϵc**, etc. Of course the path that gives the probability corresponding to the backoff equation is **abϵbc**. However, when used with the Viterbi decoding procedure, if a direct path exists between two non-null states, and its probability is higher than the probability of an indirect path, the correct probability is generally chosen. The topology described in Fig. 2 corresponds to that of the networks proposed in [3].

4. THE VARIABLE NGRAM STOCHASTIC AUTOMATON (VNSA)

In this section we provide a formalization of the Variable Ngram Stochastic Automaton (VNSA). First we will introduce stochastic non deterministic finite state automata, and then we will show that VNSA is a particular case of this class of automata.

A stochastic non-deterministic finite state automaton $\mathcal{Q}$ is described by the quintuple $\{\mathbf{S}, \mathbf{V}, \mathcal{F}, s_0, \mathbf{S}_f\}$, where $\mathbf{S}$ is a set of states s, $\mathbf{V}$ is a set of words (including the empty word ϵ) s_0 is the initial state and $\mathbf{S}_f \subset \mathbf{S}$ is a set of final states. $\mathcal{F}$ is a state transition function that given a state s and an input word $w \in V$ returns the set of pairs $\{(t_k, p_k)\}$: $\mathcal{F}(s, w) = \{(t_k, p_k)\}$, where p_k are the transition probabilities of going from state s to state t_k. The automaton starts processing from state s_0 (instant 0) and it reads the first word w_0 from the input word sequence W. Then, when in state s at the generic istant i and given an input sequence word $w_i \in \mathbf{V}$, the automaton will

1. apply state transition function to the state s with word w_i, move to the corresponding next state (if $\mathcal{F}(s, w_i) \neq \emptyset$), and read the next word in the input string, or

2. apply the state transition function to the state s with the null word ϵ, and move to the corresponding next state (the current input word is not changed).

A VNSA can be described by a non-deterministic stochastic finite state automaton. Each state $s \in S$ is associated with a m-tuple observed in the training set, $v_1, \ldots, v_m$ with $0 \leq m < n$ (n is the order of the automaton). The m-tuple $v_1, \ldots, v_m$ is called history of the state s. In the case $m = 0$ the empty word is associated to s (m is called the history size of state s). At the generic instant i, given the current sequence word w and the current state s with history $v_1, \ldots, v_m$, only two kinds of transitions are defined by the state transition function. Type 1 transition consumes an input word w and moves to a state t_j^w (non null state) with history $v_j, \ldots, v_m, w$ ($j = 1, 2$). Type 2 transition do not consume any input symbol, and moves to a state t^ϵ (null state) with history $v_2, \ldots, v_m$. The transition of type 2 is always available, while the transition of type 1 may be not. The transition to a null state corresponds to the loss of part of the state history or, in other words, to the choice of smaller history size for representing the next state in the automaton. It must be observed that in this kind of automaton all transitions leading to a non null state with history $v_1, \ldots, v_m$ recognize word v_m, while transitions leading to a null state always recognize the empty word ϵ. Moreover it has to be noticed that the predecessors of a null state s with history $v_1, \ldots v_m$ have histories of the kind $v, v_1, \ldots, v_m$, with $v \in V$. Hence all the transitions from the null state s to a non null state with smaller history size are shared by all its predecessor states.

In Fig. 3 a portion of a third order VNSA network is shown. The vocabulary of the automaton is $V = \{a, b, c, d, \epsilon\}$. For sake of clarity the transition probabilities are not shown. Null states have rectangular shape and non null states have round shape. Let us assume that the automaton is processing symbol d, while the last two symbols read are b and c. The automaton might be in state 3 or in state 8. From state 2 the automaton can step into state 4 or step into state 5. State 5 is a null state and state 1 (history ac), 2 (history dc) and 3 (history bc) can decrease the memory size through the same null state 5. Once in state 5 the automaton can release an additional memory symbol by going into null state 6, or increase the memory size by going into state 4, or keep the same memory size by stepping into state 7. From state 8 the automaton can either keep the same memory size by going into state 7 or increase it by stepping into state 4. From the analysis of Fig. 3, it is clear how the non determinism of the VNSA has been designed to cope with need of efficiency and flexibility for the language model.

Estimation of Probabilities for the VNSA

For each state s of the VNSA there is a set of word $w \in \mathbf{w}_s$ ($w \neq \epsilon$) such that $\mathcal{F}(s, w) \neq \emptyset$ and $\mathcal{F}(s, w) = \{(t_j^w, p_j^w)\}$ ($j = 1, 2$). For a state s the following equation holds:

$$\sum_{w \in \mathbf{w}_s} (p_1^w + p_2^w) + p^\epsilon = 1. \qquad (5)$$

Since state s and t_j^w are associated with m-tuples observed in the training data, a maximum likelihood estimation, $\hat{p}_j^w$, is computed via equation 2. The probability of loss of memory p^ϵ is computed through the following model for p_j^w: $p_j^w = a_j^w \hat{p}_j^w + b_j^w$. The computation of coefficients a_j^w and b_j^w is accomplished by means of modified versions of heuristic techniques given in [1], [2]. To increase the robusteness of the estimation of transition probabilities p_j^w, the concepts of *word classes* and *compound words* have been introduced. A *word class* is a set of words having semantic-syntactic similarities (e.g. the set of city names, the set of airport names, etc.). A *compound word* is a sequence of words with very strong correlation (e.g. "I'D LIKE TO", "HOW MUCH", "THANK YOU", etc.). The *word classes* and *compound words* have been defined by hand and then the training data has been tagged with the labels of *word classes* and *compound words*. Then a VNSA has been built on the basis of the filtered training data. The VNSAs built with *word classes* and *compound words* use the definitions of transition probabilities for class based language model given in [4].

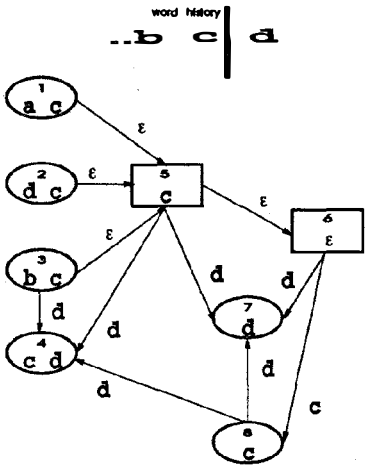

Figure 3: Portion of a 3^{rd} order VNSA network

Further considerations on the VNSA

Given an input word sequence $W = w_{\blacksquare}, w_1 \ldots, w_N$, the VNSA starts from state s_0, goes through a sequence of (null and non null states) states non null states) and ends into $s_f \in \mathbf{S_f}$. Such sequence of states ξ is called path and corresponds to a sequence of words W. There are multiple paths which are activated by the automaton for a given W. Thus the probability of W, $P(W)$, is computed through the following:

$$P(W) = \sum_{\xi \in \Xi_W} P(\xi) \prod_{s,t \in \xi} P(t|s) \qquad (6)$$

where Ξ_W is the set of all paths activated by the automaton given W, s and t are adjacent states in the sequence ξ and $P(\xi)$ is the probability of path ξ. In general the states in the sequence ξ have different memory size, thus allowing for a variable size n-gram parsing of the sequence W. This has been found useful especially with the disfluency found in ATIS database. In the case of a disfluency it is reasonable not to take into account all words previously observed in the sequence W, because they may not be able to predict the disfluency. Thus, when a disfluency is found in a sentence, the automaton generally operates at the minimum memory size.

An other important issue is state clustering in order to reduce the size of the network associated to a VNSA. Simple heuristic consideration on the observation frequency of the m-tuples in the training data can efficiently reduce the number of states by 30% with a not significant increase of the test-set perplexity.

5. SPEECH RECOGNITION RESULTS

VNSA networks have been integrated in the recognizer developed at AT&T for the 1994 ATIS evaluation.

	# of states	perplexity	word error rate
VNSA(2)	1.5k	25.35	4.7 %
VNSA(3)	20k	18.41	4.1%
VNSA*(3)	93k	18.09	3.6%

Table 1: Characteristics of three VNSA networks

The VNSA language models were trained from 20,000 ATIS sentences with a vocabulary of 1530 words. Three language models of increasing complexity were tried. Table 1 shows the number of states, the perplexity on Dec94 test-set and the speech recognition results for a $2nd$ order and $3rd$ order VNSA network (first two rows). On the third row of table 1 the results are shown for a $3rd$ order VNSA, using *word classes* and *compound words*. The best $3rd$ order VNSA yields a 25% error rate reduction with respect to the $2nd$ order network. In addition, the 2nd order VNSA compiled with word classes has the lowest perplexity, as also shown in table. Moreover the network compiled with word classes, even though it is bigger than the others, has a reduced search space in the Viterbi decoding procedure.

6. CONCLUSION

There are several advantages in using non deterministic stochastic language models for speech recognition. They offer an effective method of approaching the backoff without affecting the performance of the recognizer itself (i.e. the backoff is compiled in the network, rather than performed at run time). They allow the implementation of more general classes of language models, which can account for the flexibility needed in natural language recognition. The proposed Variable Ngram Stochastic Automaton has been shown to meet the the needs for low storage, accuracy, generalization and modelling of spontaneous speech.

REFERENCES

[1] I. H. Witten, T. C. Bell, "The zero Frequency Problem: Estimating the probabilities of novel events in adaptive text compression", *IEEE Trans. on Information Theory*, vol 37, No. 4, July 1991, pp. 1085-1093.

[2] N. Ney, U. Essen, "Estimating Small Probabilities by Leaving-One-Out", *Proc. of 3rd European Conference on Speech Communication and Technology*, EUROSPEECH 93, pp. 2239,2242, Berlin, Germany, September 1993.

[3] P. Placeway et al. "The Estimation of Powerful Language Models From Small and Large Corpora", Proc. ICASSP 1993, pp. 33-36, Minneapolis.

[4] P. Brown et al. "Class-Based n-gram Models of Natural Language", Computational Lingustics Vol. 18, No. 4, 1992, pp. 467-479.