Simulation of Fixed Length Word String Probability Distributions

Ralf Schlüter, Markus Nussbaum-Thom, and Hermann Ney

Dec. 30, 2010: Supplement to "Does the Cost Function Matter in Bayes Decision Rule?" (to appear in IEEE Transaction on Pattern Analysis and Machine Analysis).

I. Word String Distribution Simulation

The aim of this description is to provide a framework for simulating specific word string distributions whose general structure follows the structure of word string posterior distributions relevant for string recognition. For the specific case of string classes $w_1^N = w_1, w_2, \ldots, w_N$ of fixed length N with words $w_n \in \mathcal{V}$, with vocabulary size $V = |\mathcal{V}|$, and corresponding observation vectors per word position, $x_1^N = x_1, \ldots, x_N$, we consider simulations of three types of distributions to obtain word string distributions:

1) global context dependence: simulate $p(w_1^N|x_1^N)$ directly by a normalized distribution over V^N (string) classes;
2) independent emission probabilities and bigram prior: simulate $p(w|x)$ and $p(w|v)$ to obtain $p(w_1^N|x_1^N) = \prod_{n=1}^{N} p(w_n|x_n) \cdot p(w_n|w_{n-1})$;
3) complete context independence: only simulate "emission" probabilities $p(w|x_n)$ (which might already include a zerogram or unigram prior) to obtain $p(w_1^N|x_1^N) = \prod_{n=1}^{N} p(w_n|x_n)$. This case is covered by the previous case using a "bigram" with maximum perplexity, i.e. a bigram with uniform distribution over all words in all contexts: $p(w|v) = \frac{1}{V}$, with the vocabulary size $V = |\mathcal{V}|$.

The authors are with Lehrstuhl für Informatik 6 - Computer Science Department, RWTH Aachen University, Ahornstr. 55, 52074 Aachen, Germany. Emails: {schlueter,nussbaum,ney}@cs.rwth-aachen.de

In the Cases 2. and 3., the word string posterior probability distribution can be written as:

$$
\begin{aligned}
p(w_1^N|x_1^N) =& \frac{\prod_{n=1}^{N} p(x_n|w_n) \cdot p(w_n|w_{n-1})}{\sum_{v_1^N} \prod_{n=1}^{N} p(x_n|v_n) \cdot p(v_n|v_{n-1})} \\
=& \frac{\prod_{n=1}^{N} \overbrace{\frac{p(x_n|w_n)}{\sum_u p(x_n|u)}}^{=:p_n(w_n)} \cdot p(w_n|w_{n-1})}{\sum_{v_1^N} \prod_{n=1}^{N} \underbrace{\frac{p(x_n|v_n)}{\sum_u p(x_n|u)}}_{=:p_n(v_n)} \cdot p(v_n|v_{n-1})} \\
=& \frac{\prod_{n=1}^{N} p_n(w_n) \cdot p(w_n|w_{n-1})}{\sum_{v_1^N} \prod_{n=1}^{N} p_n(v_n) \cdot p(v_n|v_{n-1})}
\end{aligned}
$$

with a unigram distribution for the first word position:

$$p(w_1|w_0) := p(w_1)$$

with $w_0 := \$$ being a sentence start symbol preceding any word string.

Effectively, for each "utterance" to be simulated, as represented by an observation sequence x_1^N, it is therefore sufficient to simulate distributions $p_n(w)$ normalized over all words w instead of the original emission distributions $p(x_n|w)$ for Cases 2. and 3.

For Case 1., for each "utterance" a distribution $p(w_1^N)$ over all word strings w_1^N has to be simulated, where each word string w_1^N is considered as a completely separate class, ignoring equal words in individual positions of different strings.

In all three cases, each new simulation can be viewed as representing another sequence of observations x_1^N, i.e. the observation sequence is not generated explicitly, but implicitly by simulating its renormalized emission distribution $p_n(w)$. Although the terminology does not fit exactly, for simplicity, the simulation of $p(w_1^N)$ in Case 1., and the simulation of $p_n(w)$ for each n in Cases 2. and 3. is subsumed under the topic "emission" simulation in Sec. I-A, whereas a specific approach to simulate bigram distributions for Case 2. is presented in Sec. I-B.

A. Emission Simulation

The aim of this section is to derive algorithms to generate emission distributions for fixed length word strings that either are independent for each word position, or are globally context dependent.

In string recognition usually only a few words from the vocabulary have significant probabilities per position, unless the word error rate is large. Consequently also only a small number of word strings shows significant probability. Depending on the cost function chosen for *Bayes* decision rule, the word/string probability maximum is more or less directly connected to the error rate. Increasing the expected maximum word/string probability will decrease the expected error rate and vice versa.

In case of independent probability distributions in different word positions, we simulate this behavior by first drawing a maximum probability value from a Beta distribution. This approach was chosen to have better control over the expected error rate, as well as having a more efficient simulation of skewed distributions. The generation of the maximum probabilities is shown in the following algorithm:

Algorithm 1: Draw valid maximum probability for distribution over K events.
Input: vocabulary size K, Beta distribution parameters α, β.
Output: Beta distributed probability $\in [1/K, 1]$.
procedure GETMAX(K, α, β)
return $\frac{1}{K} + (1 - \frac{1}{K})$BETA($\alpha, \beta$) ▷ Beta distributed ▷ random number, rescaled to $[\frac{1}{K}, 1]$ ▷ (the corresponding procedure ▷ BETA is assumed to be given)
end procedure

The parameters of the Beta distribution will control both the average and the variance of the maximum probability value. The average will be connected to the average error rate, whereas the variance models the variability of the emission probabilities encountered in speech (due to, e.g., acoustic conditions in automatic speech recognition, or legibility in handwriting recognition).

Subsequently, for each word position, the distribution over the complete vocabulary is generated. To preserve the previously fixed maximum probability, this is done in an iterative manner, since the probability domain that preserves the maximum depends on the predefined maximum and the previously generated probabilities. To the authors knowledge, the resulting probability distribution will not follow a specific known type of distribution, but will usually produce a limited number of words with significant probabilities, if the maximum probability is large enough. The generation of a probability distribution with a fixed maximum probability value for a given class (word/string) is summarized in the following algorithm:

Algorithm 2: Simulate probability distribution with fixed maximum for a fixed class (word/string).
Input: maximum probability $p_{\max}$, vocabulary size K, maximum index $k_{\max}$.
Output: probability distribution p with maximum $p(k_{\max}) = p_{\max}$
procedure DISTRIBUTION($p_{\max}, K, k_{\max}$)
$p_{k_{\max}} \leftarrow p_{\max}$
$b \leftarrow 1 - p_{\max}$ ▷ balance still to be distributed
$k_{\text{final}} \leftarrow \lfloor (K-1)$RANDOM$(0,1) \rfloor + 1$ ▷ random integer $\in \{1, \ldots, K-1\}$
if $k_{\text{final}} \geq k_{\max}$ **then**
$k_{\text{final}} \leftarrow k_{\text{final}} + 1$
end if
$n \leftarrow 1$
for $k \in \{1, \ldots, K\} \backslash \{k_{\text{final}}, k_{\max}\}$ (random order) **do**
$n \leftarrow n + 1$
$l \leftarrow \max\{0, b - (K - n) * p_{\max}\}$ ▷ bounds: keep
$u \leftarrow \min\{p_{\max}, b\}$ ▷ $p_k \leq p_{\max} \forall$ remaining k
$p_k \leftarrow$ RANDOM(l, u) ▷ uniform distributed random number $\in [l, u[$.
$b \leftarrow b - p_k$ ▷ update balance
end for
$p_{k_{\text{final}}} \leftarrow b$
return p ▷ return distribution array p
end procedure

The distribution of word sequences observed during testing and the prior string distribution should ideally coincide. The emission probabilities on the other hand usually are significant for those words that are spoken in the corresponding positions in the test data. To take this into account in the simulations with independent emission distributions for each word position, the words that get assigned the maximum string emission probability in each position are drawn from the also simulated bigram prior (or unigram, at sentence start) prior distribution. The algorithm to draw a word index according to a prior distribution (unigram, or bigram for given predecessor word index) is shown below:

Algorithm 3: Draw a random word index according to a prior distribution.
Input: probability distribution lm over vocabulary.
Output: word index i.

procedure DRAWWORD(lm)
  $i \leftarrow 1$
  $c \leftarrow lm_i$
  $r \leftarrow$RANDOM$(0, 1)$ ▷ Uniform distributed random number $\in [0, 1[$.
  while $r > c$ **do**
    $i \leftarrow i + 1$
    $c \leftarrow c + lm_i$ ▷ cumulative distribution, ▷ (can also be precomputed)
  end while
  RETURN i
end procedure

Finally, the algorithm to generate an emission distribution for word strings assuming independent emission distributions for the different word positions can be provided, where the maximizing string is drawn from a bigram-based string prior:

Algorithm 4: Simulate string emission probability with independent distributions for each position. The maximizing string is drawn from a bigram-based string prior.

Input: string length N, vocabulary size V, Beta distribution parameters α, β for maximum generation, sentence start unigram $lm1$, bigram $lm2$.

Output: probability distribution over all length N word strings.

procedure INDEPEMISSION($N, V, \alpha, \beta, lm1, lm2$)
  for $n \leftarrow 1 \dots N$ **do**
    if $n = 1$ **then**
      $w \leftarrow$DRAWWORD($lm1$) ▷ draw word ▷ from unigram
    else
      $w \leftarrow$DRAWWORD($lm2_{w_{\text{pre}}}$) ▷ draw word ▷ from bigram, given the ▷ predecessor word index w_{pre}
    end if
    $w_{\text{pre}} \leftarrow w$
    $p_{\max} \leftarrow$ GETMAX(V, α, β)
    $p_n \leftarrow$ DISTRIBUTION($p_{\max}, V, w$) ▷ get emission distribution for position n, with ▷ probabilities $p_{n,v}$ for words v in position n
  end for
  return PROBTREE(p) ▷ generate (and return) tree ▷ structure representing all length N word strings ▷ & store word emission probabilities in tree arcs, ▷ compute & store string probabilities in tree leafs ▷ (procedure PROBTREE and corresponding data ▷ structure is assumed to be given)
end procedure

Another alternative is to introduce global context dependency into the word string distribution. In this case, it is sufficient to generate a globally context dependent word string posterior distribution directly. For reasons of comparability between this case and the case of no context dependence (Case 3., see above), the maximum probability in this case is drawn from a product of Beta distributions, similar to the case of position independent emission distributions. The algorithm is given below.

Algorithm 5: Simulate word string probability distribution with global context dependence, where the maximum probability is drawn from a product of N Beta distributions.
Input: string length N, vocabulary size V, Beta distribution parameters α, β for maximum generation.
Output: probability distribution over all word strings.

procedure GLOBALDEPENDENTEMISSION(N, V, α, β)
  $p_{\max} \leftarrow 1$
  for $n \leftarrow 1 \dots N$ **do**
    $p_{\max} \leftarrow p_{\max} \cdot$ GETMAX(V, α, β)
  end for
  $s_{\max} \leftarrow \lfloor V^N \cdot$RANDOM$(0,1) \rfloor + 1$ ▷ random string index $\in \{1, V^N\}$ for maximum
  $p \leftarrow$ DISTRIBUTION($p_{\max}, V^N, s_{\max}$)
  RETURN PROBTREE(p) ▷ generate and return tree ▷ structure representing all length N word strings ▷ & store string emission probabilities in tree leafs ▷ (procedure PROBTREE and corresponding data ▷ structure is assumed to be given)
end procedure

B. Bigram Simulation

The aim of this section is to derive an algorithm to simulate bigram (including sentence start unigram) distributions with given perplexity for length N word sequences. Here, an unconventional approach is presented to generate bigram distributions that exactly meet a specific target perplexity. An alternative would be to simulate bigram distributions following specific stochastic processes, which to the knowledge of the authors can not easily be done to meet specific perplexities. Therefore, such more principled simulations would have to be filtered, discarding those simulations failing to meet a specific perplexity, with especially limited efficiency for cases with low perplexity.

Assume a sequence of M independent length N word strings $W_1^M = W_1, \ldots, W_M$, with $W_m = w_{m,1}, \ldots, w_{m,N}$ and words $w_{m,n} \in \mathcal{V}$ with a vocabulary size $V = |\mathcal{V}|$. The perplexity for the sequence of word strings then can be written as:

$$PP = \frac{1}{p(W_1^M)^{\frac{1}{M \cdot N}}}. \tag{1}$$

In the limit of $M \to \infty$ we can write the perplexity as a function of the underlying probability distributions only:

$$\begin{aligned}
N \log PP &= -\frac{1}{M} \log p(W_1^M) = \frac{1}{M} \log \prod_{m=1}^{M} p(W_m) \\
&= -\frac{1}{M} \sum_{m=1}^{M} \log p(W_m) = -\sum_{W} \frac{M(W)}{M} \log p(W) \\
&\qquad \text{with } M(W) = \#\text{ occurrences of } W \\
&\overset{M\to\infty}{\longrightarrow} -\sum_{W} p(W) \log p(W) = -\sum_{w_1^N} p(w_1^N) \log p(w_1^N) \\
&= -\sum_{w_1^N} p(w_1) p(w_2^N|w_1) \big(\log p(w_1) + \log p(w_2^N|w_1) \big) \\
&= -\sum_{w_1} p(w_1) \log p(w_1) \\
&\quad -\sum_{w_1} p(w_1) \sum_{w_2^N} p(w_2^N|w_1) \log p(w_2^N|w_1) \\
&= -\sum_{w_1} p(w_1) \log p(w_1) + \sum_{w_1} p(w_1) E(w_1)
\end{aligned} \tag{2}$$

with the definition of the (unnormalized) conditional log perplexity (or entropy) E_N:

$$E_N(w_1) := -\sum_{w_2^N} p(w_2^N|w_1) \log p(w_2^N|w_1) \tag{3}$$

Assuming a bigram distribution, $E_N(w)$ solely depends on the bigram probabilities $p(w|v)$:

$$E_N(w_1) = -\sum_{w_2^N} \left(\prod_{n=2}^{N} p(w_n|w_{n-1}) \right) \sum_{n'=2}^{N} \log p(w_{n'}|w_{n'-1}).$$

Now assume the simplest non-trivial case of $N = 2$. In this case, Eq. 2 reduces to:

$$2 \log PP = F + G, \tag{4}$$

with the unigram log perplexity (entropy)

$$F := -\sum_{w_1} p(w_1) \log p(w_1),$$

and the average bigram conditional log perplexity (entropy)

$$\begin{aligned}
G &:= \sum_{w_1} p(w_1) E_2(w_1) \\
&= -\sum_{w_1} p(w_1) \sum_{w_2} p(w_2|w_1) \log p(w_2|w_1).
\end{aligned}$$

Utilizing the structure of Eq. (4), the simulation of the bigram and the (sentence start) unigram probability distributions, to obtain a predefined perplexity in case of $N = 2$, then can be done in the following way:

1) Draw admissible target $\hat{G}$, given PP.
2) Set target $\hat{F} = 2 \log PP - \hat{G}$.
3) Draw unigram distribution $p(w)$ such that $F = \hat{F}$.
4) Draw target $\hat{E}_2(w_1)$ for all w_1 such that $G = \hat{G}$.
5) For each w_1: Generate bigram distribution $p(w_2|w_1)$ such that $E_2(w_1) = \hat{E}_2(w_1)$.

For Steps 1) and 2), $\hat{F}$ and $\hat{G}$ need to be constrained to $[0, \log V]$, since this is the co-domain of both F and G. Further, Eq. (4) needs to be fulfilled. Therefore, for given PP, we have:

$$\hat{G} \in \big[\max\{2 \log PP - \log V, 0\}, \min\{2 \log PP, \log V\}\big],$$

and $\hat{G}$ will be drawn from a uniform distribution over this interval.

For steps 3) and 5), we need to generate probability distributions with probabilities $p := (p_1, \ldots, p_V)$ with a given entropy:

$$H(p) = -\sum_{j=1}^{V} p_j \log p_j.$$

Assuming $p_V < 1$, the entropy can be decomposed in the following way:

$$\begin{aligned} H(p) &= -p_V \log p_V - (1 - p_V)\big[\log(1 - p_V) - H(q_1^{V-1})\big] \\ &=: f\big(p_V, H(q_1^{V-1})\big) \end{aligned} \tag{5}$$

with renormalized probability distribution:

$$q_j = \frac{p_j}{1 - p_V} \quad \forall\, j = 1, \ldots, V - 1.$$

Eq. (5) can now be used to obtain the domain $I(\hat{H}_V, V)$ for p_V that allows to still meet the target entropy $H(p) = \hat{H}_V$, considering the co-domain of the entropy of the remaining distribution $H(q_1^{V-1}) \in [0, \log(V - 1)]$. The admissible domain can then be written as:

$$I(\hat{H}_V, V) = \big\{p \in [0, 1] | \exists h \in [0, \log(V - 1)] : f(p, h) = \hat{H}_V\big\},$$

It can be shown that $f(p, \hat{H})$ as a function of p is convex, therefore I is a single interval, or consists of two sub intervals, and the interval boundaries can be found by numeric solution of $f(p, \hat{H}_{V-1}) = \hat{H}_V$ (if existing) for the two extremal values of $\hat{H}_{V-1} \in \{0, \log(V - 1)\}$, and by considering the boundaries $p \in \{0, 1\}$.

This process can then be iterated to obtain a distribution with target entropy $\hat{H}_V$:

Algorithm 6: Draw probability distribution with target entropy $\hat{H}_V$.
Input: vocabulary size V, and target entropy $\hat{H}_V$.
Output: probability distribution over V words with entropy $\hat{H}_V$.

procedure TARGETENTROPYDISTRIBUTION($V, \hat{H}_V$)
  $h = \hat{H}_V$
  for $j \leftarrow V \ldots 2$ **do**
    $p_j \leftarrow$ UNIFORM$\big(I(h, j)\big)$
    if $p_j < 1$ **then**
      $h \leftarrow \frac{p \log p + (1-p) \log(1-p) + h}{1-p}$
    else
      for $i \leftarrow j - 1, \ldots, 2$ **do**
        $p_i \leftarrow 0$
      end for
      $j \leftarrow 2$
    end if
  end for
  $p_1 \leftarrow 1 - \sum_{j=2}^{V} p_j$
  PERMUTE(p) ▷ random permutation of array p

▷ (assumed to be given)
return p ▷ return distribution array p
end procedure

For Step 4), the conditional bigram log perplexities $E_2(w)$ have to be drawn for each w. For notational simplicity, we rewrite Eq. (4) by replacing $E_2(w)$ by c_j and $p(w)$ by p_j, where j enumerates the words of the vocabulary, with the vocabulary size $V = |\mathcal{V}|$:

$$\sum_{j=1}^{V} p_j c_j = \hat{G}. \tag{6}$$

Now we have to draw the set $\{c_j | j = 1, \ldots, V\}$ under the constraints $0 \leq c_j \leq \log V$ that fulfills Eq. (6) for a given probability distribution p_j. This process will be done iteratively. Assume that $c_j = \hat{c}_j$ already were drawn for $j = 1, \ldots, i-1$. Then Eq. (6) can be rewritten to:

$$p_i c_i + \sum_{j=i+1}^{V} p_j \cdot c_j = \hat{G} - \sum_{j=1}^{i-1} p_j \cdot \hat{c}_j =: \hat{G}_i.$$

In case of $p_i = 0$, c_i does not contribute to Eq. (6) and can therefore be drawn from the interval $[0, \log V]$. Define $Z(i)$ as the number of non-zero probabilities in p_{i+1}^V; if all p_{i+1}^V are non-zero, we have $Z(i) = V - i$. Then, for $p_i > 0$, the admissible interval for c_i is:

$$c_i \in \left[\max\left\{0, \frac{\hat{G}_i - Z(i) \log V}{p_i}\right\}, \min\left\{\frac{\hat{G}_i}{p_i}, \log V\right\}\right].$$

With this, the algorithm to draw the conditional bigram log perplexities can then be done iteratively:

Algorithm 7: Draw conditional bigram log perplexities that meet given average $\hat{G}$ over given probability distribution p.

procedure BIGRAMLOGPERPLEXITIES($V, \hat{G}, p$)
  $g \leftarrow \hat{G}$
  $z \leftarrow \sum_{\substack{j=1 \\ p_j > 0}}^{V} 1$
  for $j \in \{1, \ldots, V\}$ in random order **do**
    if $p_j > 0$ **then**
      $z \leftarrow z - 1$
      $c_{\min} = \max\{0, (g - z \log V)/p_j\}$
      $c_{\max} = \min\{g/p_j, \log V\}$
      $c_j \leftarrow \text{UNIFORM}\big(c_{\min}, c_{\max}\big)$
      $g \leftarrow g - p_j c_j$
    else
      $c_j \leftarrow \text{UNIFORM}\big(0, \log V\big)$
    end if
  end for
  return c ▷ return array c of log perplexities
end procedure

This concludes the simulation of a bigram probability distribution in the case of length two strings. For longer strings, the simulation cannot be done as easily, cf. (2). Nevertheless, under certain conditions, assuming a given bigram distribution, the variation of the corresponding sentence start unigram distribution can lead to a given length N string perplexity. First of all, note that the conditional log perplexity $E_N(w)$ for each context w is solely defined by the (in case of $N > 2$, complete) bigram distribution, cf. Eq. (3). The co-domain of the conditional log perplexities is $[0, (N-1) \log V]$. Again, we replace $p(w)$ by p_j, and $E_N(w)$ by a_j with $a = (a_1, \ldots, a_V)$ to simplify notation. Now define the following function:

$$f(p) = -\sum_{j=1}^{V} p_j \log p_j + \sum_{j=1}^{V} p_j a_j,$$

with the lower bound

$$f(p) \geq \min_i a_i = f(q)$$

with $q = (q_1, \ldots, q_V)$ and

$$q_j = \begin{cases} 1 & \text{for } j = \arg\min_i a_i \\ 0 & \text{otherwise.} \end{cases}$$

The upper bound can be derived using *Gibbs*' Inequality:

$$f(p) \leq \log \sum_{i=1}^{V} \exp(a_i) = f(\rho)$$

with $\rho = (\rho_1, \ldots, \rho_V)$ and

$$\rho_j = \frac{\exp(a_j)}{\sum_{i=1}^{V} \exp(a_i)} \text{ for } j \in \{1, \ldots, V\}.$$

For given $a_j \in [0, (N-1)\log V]$ the aim now is to modify the unigram distribution p such that

$$f(p) = N \log PP \tag{7}$$

is fulfilled. This only is possible if the target log perplexity is within the bounds derived above:

$$\min_{j=1,\ldots,V} a_j \leq N \log PP \leq \log \left(\sum_{j=1}^{V} \exp(a_j) \right). \tag{8}$$

If the Ineqs. (8) are not fulfilled, a new initial unigram and bigram are generated, until the condition is fulfilled. In practice, only few iterations (if any) where necessary.

To fulfill Eq. (7), the modified distribution is obtained by linear interpolation between an initial distribution p and either the minimizing distribution q or the maximizing distribution ρ and numerically solving Eq. (7) w.r.t. the interpolation parameter. The function $f(p_1, \ldots, p_V)$ can be shown to be convex, therefore the solution can be obtained numerically by nested intervals, as shown in the following algorithm:

Algorithm 8: Modify sentence start unigram distribution for given bigram distribution to fit perplexity to a given target length N string perplexity.

Input: string length N, vocabulary size V, target perplexity PP, initial distribution p, array of a of conditional length $N-1$ log perplexities (entropies).

Output: modified distribution p, or empty set in case of failure.

procedure FITSTRINGPERPLEXITY(N, V, PP, p, a)
  $\epsilon \leftarrow 10^{-8}$
  $A_{\min} \leftarrow \min_i a_i$
  $A_{\max} \leftarrow \log \sum_{i=1}^{V} \exp(a_i)$
  $A \leftarrow 0$
  for $i \leftarrow 1 \ldots V$ **do**
    $q_i \leftarrow 0$
    $\rho_i \leftarrow \exp(a_i)/\exp(PP_{\max})$
    $A \leftarrow A - p_i \log p_i + p_i a_i$
  end for
  $q_{\arg\min_i a_i} \leftarrow 1$
  if $A < A_{\min}$ OR $A > A_{\max}$ **then**
    RETURN $\emptyset$
  end if
  if $A > N \log PP$ **then**
    $\rho \leftarrow p$
  else
    $q \leftarrow p$
  end if
  while $\sum_{i=1}^{V} |\rho_i - q_i| > \epsilon$ **do**
    $A \leftarrow 0$
    for $i \leftarrow 1 \ldots V$ **do**
      $p_i \leftarrow (\rho_i + q_i)/2$
      $A \leftarrow A - p_i \log p_i + p_i a_i$
    end for
    if $A > N \log PP$ **then**
      $\rho \leftarrow p$
    else
      $q \leftarrow p$
    end if
  end while
  RETURN p
end procedure

The algorithm can also be done without using an initial distribution p, which would result in a unique solution given the conditional log perplexities for length N strings. Nevertheless, here, the algorithm is initialized with p as obtained from string length $N = 2$ bigram/unigram generation, to introduce variability. Initialization with random p also is possible and was tested as well, it allows to exploit all possible unigram distributions that lead to the target perplexity.

Finally, the above procedures can be integrated into an algorithm to generate a bigram (including sentence start unigram) probability distribution that leads to a given length N perplexity:

Algorithm 9: Generation of a bigram distribution (including sentence start unigram distribution) with a fixed perplexity for length N word strings and a vocabulary size V.
Input: word string length N, vocabulary size V, target string (length N) perplexity PP.
Output: bigram and sentence start unigram probability distributions $lm2$ and $lm1$ giving length N string perplexity PP.

procedure GETBIGRAM((N, V, PP))
  repeat
    $G_{\min} \leftarrow \max\{2 \log PP - \log V, 0\}$
    $G_{\max} \leftarrow \min\{2 \log PP, \log V\}$
    $G \leftarrow$RANDOM$(G_{\min}, G_{\max})$
    $F \leftarrow 2 \log PP - G$
    $lm1 \leftarrow$TARGETENTROPYDISTRIBUTION(V, F)
    $c \leftarrow$BIGRAMLOGPERPLEXITIES$(V, G, lm1)$
    for $i \leftarrow 1, \ldots, V$ **do**
      $lm2_i \leftarrow$TARGETENTROPYDISTRIBUTION(V, c_i)
    end for
    $a \leftarrow$CONDITIONALENTROPIES$(N, lm2)$
    ▷ Not provided here: Efficient computation of
    ▷ length N conditional bigram entropies, cf.
    ▷ Eq. (3), on tree of all length N word strings.
    $lm1 \leftarrow$FITSTRINGPERPLEXITY$(N, V, PP, lm1, a)$
  until $lm1 \neq \emptyset$
  RETURN $lm2, lm1$
end procedure

For efficient computation, the bigram and unigram probability then are stored at the arcs and the roots respectively of a tree over all possible length N word strings, as already motivated for the emission probabilities in Algorithm 4 in Sec. I-A.