PUTTING LANGUAGE INTO LANGUAGE MODELING[†]

Frederick Jelinek *Ciprian Chelba*

Center for Language and Speech Processing
The Johns Hopkins University, Baltimore, MD-21218, USA
{jelinek,chelba}@jhu.edu

ABSTRACT

In this paper we describe the statistical Structured Language Model (SLM) that uses grammatical analysis of the hypothesized sentence segment (prefix) to predict the next word. We first describe the operation of a basic, completely lexicalized SLM that builds up partial parses as it proceeds left to right. We then develop a chart parsing algorithm and with its help a method to compute the prediction probabilities $P(w_{i+1}|\mathbf{W}_i)$. We suggest useful computational shortcuts followed by a method of training SLM parameters from text data. Finally, we introduce more detailed parametrization that involves non-terminal labeling and considerably improves smoothing of SLM statistical parameters. We conclude by presenting certain recognition and perplexity results achieved on standard corpora.

1. INTRODUCTION

In the accepted statistical formulation of the speech recognition problem [1] the recognizer seeks to find the word string

$$\widehat{\mathbf{W}} \doteq \arg\max_{\mathbf{W}} P(\mathbf{A}|\mathbf{W})\, P(\mathbf{W})$$

where $\mathbf{A}$ denotes the observable speech signal, $P(\mathbf{A}|\mathbf{W})$ is the probability that when the word string $\mathbf{W}$ is spoken, the signal $\mathbf{A}$ results, and $P(\mathbf{W})$ is the a priori probability that the speaker will utter $\mathbf{W}$.

The language model estimates the values $P(\mathbf{W})$. With $\mathbf{W} = w_1, w_2, \ldots, w_n$ we get by Bayes' theorem,

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i|w_1, w_2, \ldots, w_{i-1})$$

Since the parameter space of $P(w_i|w_1, w_2, \ldots, w_{i-1})$ is too large[1], the language model is forced to put the *history* $\mathbf{W}_{i-1} = w_1, w_2, \ldots, w_{i-1}$ into an equivalence class determined by a function $\Phi(\mathbf{h})$. As a result,

$$P(\mathbf{W}) \cong \prod_{i=1}^{n} P(w_i|\Phi(\mathbf{W}_{i-1})) \qquad (1)$$

Research in language modeling consists of finding appropriate equivalence classifiers Φ and methods to estimate $P(w_i|\Phi(\mathbf{W}_{i-1}))$.

The language model of state-of-the-art speech recognizers uses $(N-1)$-gram equivalence classification, that is, defines

$$\Phi(\mathbf{W}_{i-1}) \doteq w_{i-N+1}, w_{i-N+2}, \ldots, w_{i-1}$$

Once the form $\Phi(\mathbf{W}_{i-1})$ is specified, only the problem of estimating $P(w_i|\Phi(\mathbf{W}_{i-1}))$ from training data remains.

In most cases, $N = 3$ which leads to a *trigram* language model. The latter has been shown to be surprisingly powerful and, essentially, all attempts to improve on it in the last 20 years have failed. The one interesting enhancement, facilitated by maximum entropy estimation methodology, has been the use of *triggers* [2] or *singular value decomposition* [3] (either of which dynamically identify the topic of discourse) in combination with $N-$gram models .

2. GRAMMATICAL ANALYSIS OF THE HISTORY

It has always seemed desirable to base the language model on an equivalence classifier Φ that would take into account the *grammatical structure* of the history $\mathbf{h}$. Until very recently all attempts to do so have faltered. The causes of failure were probably (a) the left-to right requirement for a language model, (b) inadequate parametrization, and (c) sparseness of data.

Fortunately, we have had some initial success with the *Structured Language Model* (SLM) [4], [5],[12] that both reduces entropy and the error rate. In this presentation we give a description of operation of a basic SLM (Section 3), discuss its training, provide a new parsing algorithm, generalize the basic approach, and conclude by reviewing the results achieved so far.

It should be stressed that the development of the SLM is in its infancy, and that with further study we expect considerable progress, particularly by improved parametrization and shortcuts in training.

Finally, it is worth mentioning that cursory inspection of the structural analysis provided by the SLM indicates the possibility of its use as a general parser. We plan to pursue this avenue of research in the future.

3. A SIMPLE STRUCTURED LANGUAGE MODEL

In this section we will describe the simplest SLM. It is completely lexical. That is, phrases are annotated by headwords but not by non-terminals. The text itself is not tagged. Because of the necessarily sparse (relative to what is required for the task) amount of data from which its parameters would be estimated, a practical SLM would require non-terminal annotation. An SLM "complete" in this sense is described in [4] and we will discuss it in Section 9.

In the following description we act as if the words w_i of a sentence are fed in sequence to the SLM which makes definite probability-based decisions concerning its actions. But a language model needs to operate as in (1). We will therefore eventually need to give an algorithm computing

$$P(w_i|\Phi(\mathbf{W}_{i-1})) = \sum_{\mathbf{T}_i} P(w_i, \mathbf{T}_{i-1}|\mathbf{W}_{i-1}) \qquad (2)$$

where the sum is over all the possible structures $\mathbf{T}_{i-1}$ assigned by the SLM to the history $\mathbf{W}_{i-1}$. Such an algorithm is given in Section 6 and its more practical version in Section 7.

As the operation of the SLM proceeds a sentence and its parse are generated. The parse consists of a binary tree whose nodes are marked by *headwords* of phrases spanned by the subtree

[†]THIS WORK WAS FUNDED BY THE NSF IRI-19618874 GRANT STIMULATE

[1]The words w_j belong to a vocabulary $\mathcal{V}$ whose size is in the tenths of thousands.

stemming from the node. The headword at the apex of the final tree is $< s >$. The operation of the basic SLM is based on *constructor* moves and *predictor* moves. The headword of a phrase can be any word belonging to the span of the phrase.

1. **Constructor moves**:

   The constructor looks at the pair of right-most *exposed* headwords[2], h_{-2}, h_{-1} and takes an *action* a with probability $Q(a|\ h_{-2}, h_{-1})$ where $a \in \{$adjoin **right**, adjoin **left**, **null**$\}$. The definitions of the three possible actions are:

   - adjoin **right**: Create an apex marked by the identity of h_{-1} and connect it by a leftward branch to the (formerly) exposed headword h_{-2} and by a rightward branch to the exposed headword h_{-1} (i.e., the headword h_{-1} is percolated up by one tree level). Increase the indices of the current exposed headwords $h_{-3}, h_{-4}, \ldots$ by 1. These headwords together with h_{-1} become the new exposed headwords $h'_{-1}, h'_{-2}, h'_{-3}, \ldots$. I.e., $h'_{-1} = h_{-1}$, and $h'_{-i} = h_{-i-1}$ for $i = 2, 3, \ldots$.
   - adjoin **left**: Create an apex marked by the identity of h_{-2} and connect it by a leftward branch to the (formerly) exposed headword h_{-2} and by a rightward branch to the exposed headword h_{-1} (i.e., the headword h_{-2} is percolated one tree level up). Increase the indices of the new apex, as well as those of the current exposed headwords $h_{-3}, h_{-4}, \ldots$ by 1. These headwords thus become the new exposed headwords $h'_{-1}, h'_{-2}, h'_{-3}, \ldots$. I.e., $h'_{-i} = h_{-i-1}$ for $i = 1, 2, 3, \ldots$.
   - **null**: Leave headword indexing and current parse structure as they are and pass control to the predictor.

   If $a \in \{$adjoin **right**, adjoin **left**$\}$ the constructor stays in control and chooses the next action with probability $Q(a|h_{-2}, h_{-1})$ where the latest (possibly newly created) headword indexation is always used. If $a =$ **null**, the constructor suspends operation and the control is passed to the *predictor*.

   Note that a **null** move means that in the eventual parse the presently right-most exposed headword will be connected to the right. The adjoin moves connect the right-most exposed headword to the left.

2. **Predictor moves**:

   The predictor generates the next word w_j with probability $P(w_j = v|h_{-2}, h_{-1}),\ v \in \mathcal{V} \cup < |s >$. The indexing of the current headwords $h_{-1}, h_{-2}, h_{-3}, \ldots$ is decreased by 1 and the newly generated word becomes the right-most exposed headword. Thus $h'_{-1} = w_j$, $h'_{-i} = h_{-i+1}$ for $i = 2, 3, \ldots$. Control is then passed to the constructor.

The operation ends when the SLM completes the tree by marking its apex by the headword $< s >$.

To complete the description of the operation of the SLM, we have to take care of initial conditions:

Start of operation: The predictor generates the first word w_1 with probability

$$P_1(w_1 = v) = P(w_1 = v| < s >),\ v \in \mathcal{V}$$

The initial headwords (both exposed) become $h_{-2} =< s >$, $h_{-1} = w_1$. Control is passed to the constructor.

[2] A headword is exposed if, *at the time in question*, it is not the progeny of another headword, i.e., if it is not (yet) part of a phrase with a head of its own.

Special constructor probabilities:

$$Q(a|h_{-2} =< s >, h_{-1} \neq < |s >) = \begin{cases} 1, & a = \mathbf{null} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

$$Q(a|h_{-2} \neq < s >, h_{-1} =< |s >) = \begin{cases} 1, & a = \mathbf{right} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

$$Q(a|h_{-2} =< s >, h_{-1} =< |s >) = \begin{cases} 1, & a = \mathbf{left} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

Special predictor probabilities:

$$P(< |s > \ | h_{-2} \neq < s >, h_{-1}) = 0 \quad (6)$$

It should be noted that requirement (6) allows the end of sentence marker $< |s >$ to be generated only if the parse is ready for completion when there are only two exposed headwords, the first of which is the beginning of sentence marker $< s >$ and the second is an "ordinary" lexical headword h_{-1}. Once $< |s >$ is generated, rules (4) and (5) are applied in succession thereby completing the parse. Note that rule (3) allows the generation of w_{-2} while preventing the joining of w_0 and w_1 into a phrase.

An example of a final parse of the sentence "THE LANGUAGE MODEL ESTIMATES THE VALUES $P(\mathbf{W})$" is shown in Figure 1, and Figure 2 shows its development (a sub–parse) just before the second THE is generated (note in particular the exposed heads $h_{-1} =$ ESTIMATES, $h_{-2} =$ MODEL, $h_{-3} = < s >$).

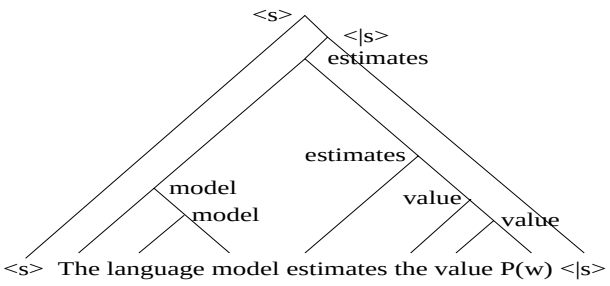

Figure 1. Complete Parse

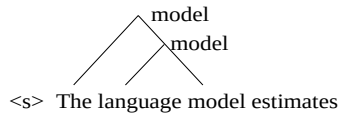

Figure 2. Partial Parse

Let $a_{i,1}, a_{i,2}, \ldots, a_{i,k_i}$ be the actions taken by the constructor when it is presented with the history $\mathbf{W}_i$. Necessarily, $a_{i,k_i} =$ **null** and $a_{i,j} \in \{\mathbf{left}, \mathbf{right}\}$ for $1 \leq j < k_i$. Then

$$P(\mathbf{T}, \mathbf{W}) = \prod_{i=1}^{n+1} P(w_i|\mathbf{h}(\mathbf{T}_{i-1})) \prod_{j=1}^{k_i} Q(a_{i,j}|\mathbf{h}(\mathbf{T}_i)) \quad (7)$$

where $\mathbf{T}_i$ denotes the partial parse constructed on $\mathbf{W}_i$, including its headwords, and $\mathbf{h}(\mathbf{T}_i)$ denotes the two most recent exposed headwords h_{-1} and h_{-2} of the partial parse $\mathbf{T}_i$. Of course, $\mathbf{T}_{n+1} = \mathbf{T}$.

4. THE LANGUAGE MODEL

As pointed out in Section 3, we must now show how the SLM can be used to compute the language model probabilities (2)

$$P(w_i|\Phi(\mathbf{W}_{i-1})) = \sum_{\mathbf{T}_i} P(w_i, \mathbf{T}_{i-1}|\mathbf{W}_{i-1})$$

To do so, we will first develop a chart parsing algorithm [6],[7],[8]. In a previous paper [4] we have shown how to approximate the summation in (2) with the help of stacks that hold as entries the dominant terms $P(w_i, \mathbf{T}_{i-1}|\mathbf{W}_{i-1})$ of that sum. The chart parsing algorithm is, of course, of interest in its own right since the SLM may be used simply as a parser. The algorithm will also lead to a Viterbi-like determination of the most probable parse (see Section 8)

$$\widehat{\mathbf{T}} = \arg\max_{\mathbf{T}} P(\mathbf{T}, \mathbf{W}) \tag{8}$$

5. A CHART PARSING ALGORITHM

We now proceed under our simplified assumption that the SLM operates on words only and does not use either tags or non-terminals. We will derive a recursion (see (10)) that can be used to calculate $P(\mathbf{W})$.

As before, $\mathbf{W}$ denotes a string of words $w_0, \ldots, w_{n+1}$ that form the complete sentence, where w_i, $i = 1, \ldots, n$ are elements of a vocabulary $\mathcal{V}$, $w_0 =< s >$ (the beginning of sentence marker, generated with probability 1) and $w_{n+1} =< /s >$ (the end of sentence marker). The first word, w_1 is generated with probability $P_1(w_1) = P(w_1| < s >)$, the rest with probability $P(w_i|h_{-2}, h_{-1})$ where h_{-2}, h_{-1} are the most recent exposed headwords valid at the time of generation of w_i. The algorithm we will develop will be computationally quite complex because the exposed headword pairs $\mathbf{h}(\mathbf{T}_i)$ determine the parser's moves, and as $\mathbf{T}_i$ varies, $\mathbf{h}(\mathbf{T}_i)$ can be any word pairs w_j, w_l, $0 \le j < l \le i$ belonging to the prefix $\mathbf{W}_i$.

Let

$$xy[i,j] \doteq P(\mathbf{w}_{i+1}^j, h(\mathbf{w}_i^j) = y|h_{-1}(\mathbf{w}_0^{i-1}) = x, w_i),$$

$1 \le i < j < n+1$, denote the probability that, given that x is the last exposed headword preceding time i and that w_i is generated, the following words $\mathbf{w}_{i+1}^j = w_{i+1} \ldots, w_j$ are generated, $\mathbf{w}_i^j = w_i, w_{i+1} \ldots, w_j$ becomes a phrase and y is its headword.

Define further the boundary conditions

$$xy[i,j] \doteq 0, \text{ if } x \notin \{w_0, \ldots, w_{i-1}\} \text{ or } y \notin \{w_i, \ldots, w_j\} \text{or } i > j$$

and, for $j = 1, 2, \ldots, n$,

$$xy[j,j] \doteq \begin{cases} 1 & \text{for } x \in \{w_0, \ldots, w_{j-1}\}, y = w_j \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Then,[3] for $1 \le i < j < n+1$,

$$\begin{aligned} xy[i,j] = & \qquad (10) \\ & \sum_{l=i}^{j-1} \sum_z xy[i,l]\ P^*(w_{l+1}|x,y)\ yz[l+1,j]\ Q(\mathbf{left}|y,z) \\ + & \sum_{l=i}^{j-1} \sum_v xv[i,l]\ P^*(w_{l+1}|x,v)\ vy[l+1,j]\ Q(\mathbf{right}|v,y) \end{aligned}$$

where

$$P^*(w|h_{-2}, h_{-1}) = P(w|h_{-2}, h_{-1})\ Q(\mathbf{null}|h_{-2}, h_{-1})$$

The probability we are interested in is then given by

$$P(\mathbf{W}) = \ w_0 w_{n+1}[1, n+1] \tag{11}$$

[3] For justification see the two paragraphs following (11).

To justify formula (10), observe the following: one of the ways to generate $w_{i+1}, \ldots, w_j$ and create a phrase spanning $[i,j]$ whose headword is y, given that the headword of the preceding phrase is x and the word w_i was generated, is that there is a string $w_{i+1}, \ldots, w_l$ generated, that a phrase spanning $[i,l]$ is formed whose headword is y (and preceding that phrase is another one whose headword is x), that the word w_{l+1} is generated from its two preceding headwords (i.e., x, y), that the string $w_{l+2}, \ldots, w_j$ is generated and the span $[l+1, j]$ forms a following phrase whose headword is, say, z (and the headword of its preceding phrase must be y!) and that the two phrases are joined as one whose headword is y.

Another way to create a phrase whose headword is y and to generate $w_{i+1}, \ldots, w_j$, given that the headword of the preceding phrase is x and the word w_i was generated, is almost identical, except that the first of the two phrases is headed by some headword v and the second by headword y, and when these two phrases are joined it is the second headword, y, which is percolated upward to head the overall phrase. Of course, in this case w_{l+1} is generated from its preceding two headwords, x and v.

Our chart algorithm will proceed left-to-right,[4] starting with $w_0 w_1[1,1] = 1$. The probabilities of phrases covering word position spans $[i,j]$, $i < j$ will be calculated from (10) after the corresponding information concerning spans $[k, j-1]$, $k = 1, \ldots, j-2$ and $[l,j]$, $l = i+1, \ldots, j-1$ had been determined.[5]

6. COMPUTING $P(W_{I+1}|\mathbf{W}_I)$

We can now use the concepts and notation developed in the preceding section to compute left-to-right probabilities of word generation by the SLM. Let $x[i]$ denote the probability that the sequence $w_0, w_1, w_2, \ldots, w_i, w_{i+1}$ is generated and the partial parse of $\mathbf{W}_i$ is any subtree $\mathbf{T}_i$ whose last exposed headword is x.[6] Further, define the set of words $\mathcal{W}^i = \{w_0, w_1, w_2, \ldots, w_i\}$. Then we have for $l = 1, 2, \ldots, n$ the following recursion:

$$x[l] = \sum_{i=0}^{l-1} \sum_{y \in \mathcal{W}^i} y[i]\ \ yx[i+1, l]\ P^*(w_{l+1}|y,x) \tag{12}$$

for $x \in \{w_1, \ldots, w_l\}$, with the initial condition

$$x[0] = \begin{cases} P_1(w_1) & x = w_0 \\ 0 & x \ne w_0 \end{cases}$$

It follows directly from (12) that for $i = 1, 2, \ldots, n$

$$P(w_0, w_1, w_2, \ldots, w_i, w_{i+1}) = \sum_{x \in \mathcal{W}^i} x[i]$$

and therefore

$$P(w_{i+1}|w_0, w_1, w_2, \ldots, w_i) = \frac{\sum_{x \in \mathcal{W}^i}\ x[i]}{\sum_{y \in \mathcal{W}^{i-1}}\ y[i-1]} \tag{13}$$

It follows that to calculate $P(w_i|w_0, w_1, w_2, \ldots, w_{i-1})$ we must have had in our possession the values $x[j]$, $j =$

[4] So can the famous CYK algorithm [6],[7],[8] that is similar to but simpler than ours. As a matter of fact, it is obvious from formula (10) that the presented algorithm can also be run from bottom up, but such a direction would be computationally wasteful as indicated in Section 7 that discusses computational shortcuts.

[5] Note from (9) that the values $xy[j,j]$ are known.

[6] That is, $\mathbf{T}_i$ is a subtree "covering" the prefix $\mathbf{W}_i = w_0, w_1, w_2, \ldots, w_i$, the constructor passes control to the predictor which then generates the next word w_{i+1}. Thus

$$x[i] = \sum_{\mathbf{T}_i} P(\mathbf{W}_i, h_{-1}(\mathbf{T}_i) = x, w_{i+1})$$

$0, 1, \ldots, i-1$, and the values $xy[l,j]$ for $1 \leq l < j \leq i-1$ for the appropriate combinations of $x, y \in \mathcal{W}^{i-1}$. To then calculate $P(w_{i+1}|w_0, w_1, w_2, \ldots, w_i)$ we must first calculate the values $xy[l,i]$ for $1 \leq l < i$ and with their help $x[i]$ for $x, y \in \mathcal{W}^i$.

7. LIMITING THE EFFORT IN CALCULATING $P(W_{I+1}|\mathbf{W}_I)$

It is the nature of the SLM that, with positive probability, a phrase spanning $[i,j]$ can have as its headword any of the words $\{w_i, \ldots, w_j\}$. As a result, the computational complexity of the algorithms of the preceding two sections is proportional to n^6. This makes these algorithms impractical, unless a scheme can be devised that would purge from the chart a substantial fraction of its entries.

Observe first that the number of constructor moves creating any particular binary tree spanning $[i,j]$ is constant,[7] and that the number of different binary trees that can span $[i,j]$ is also constant. Therefore, the values of the probabilities $xy[i,j]$ are comparable to each other regardless of the identity of $x \in \{w_0, w_1, \ldots, w_{i-1}\}$ and $y \in \{w_i, \ldots, w_j\}$. They can thus be thresholded with respect to $\max_{x,y} xy[i,j]$.

It must, of course, be kept in mind that thresholding is only an opportunistic device: The fact that $xy[i,j] << \max_{v,z} vz[i,j]$ does not mean that $xy[i,j]$ cannot be part of some highly probable parse, since, for instance, $yz[j+1,k]$ may be very large and thus compensate for the relatively small value of $xy[i,j]$. That is, the headword y might be "needed" to complete the parse $xz[i,k]$. Similarly, the value of $vx[m,i-1]$ could be very large which would make the phrase $vy[m,j]$ attractive, again, in spite of the smallness of $xy[i,j]$.

Next note that if $x[k] << \max_z z[k]$ then it is unlikely that a high probability parse will account for the interval $[0,k]$ with a sub-parse whose last headword is x. In such a case then, the calculation of $xy[k+1,j]$, $j \in \{k+2, \ldots, n+1\}$ will probably not be needed (for any y) because in (10) and (12) $xy[k+1,j]$ will be multiplied by $vx[i,k]$ and $x[k]$, respectively.

Again, the fact that $x[k]$ is small does not mean that the headword x cannot be useful in producing the future. I.e., it is still possible (though unlikely) that $xy[k+1,j]$ for some y and j will be so large that at least some parses over the interval $[0,j]$ having x as the last exposed headword at time k will correspond to a substantial probability mass.

Should we wish to take advantage of the thresholding opportunities inherent in the above observations, we ought to compute the probabilities (10) and (12) in the following sequence: once $x[i]$ and $xy[j,i]$, $j = 1, 2, \ldots, i-1$, $i = 1, 2, \ldots, l$ are known, probabilities $vz[k, l+1]$ are computed in the sequence $k = l, l-1, \ldots, 1$. Equation (12) then allows us to compute $x[l+1]$ for the various headwords x and the cycle continues. During this computation, thresholding mentioned in the preceding two paragraphs is carried out.

Thresholding has certain unpleasant consequences. In particular, if in order to save on computation some small quantities are set to 0 then the quantity defined by (13) will no longer be a probability since we can not guarantee that it will be normalized. To assure proper normalization, we can proceed as follows. Define

$$Q_l(y,x) \doteq Q(\mathbf{null}|y,x) \sum_{i=0}^{l-1} y[i]\ yx[i+1,l] \qquad (14)$$

where the terms in the sum are those obtained after thresholding, if any. Further let $\mathcal{S}_l = \{y, x : Q_l(y,x) > 0\}$. Then

$$P^-(w_{i+1}|w_0, w_1, w_2, \ldots, w_i) = \frac{1}{K_i} \sum_{y,x \in \mathcal{S}_i} Q_i(y,x)\ P(w_i|y,x) \qquad (15)$$

[7] In fact, exactly $j-i$ adjoint moves and $j-i$ null moves are needed to construct a binary tree spanning $[i,j]$.

is a proper probability with

$$K_l = \sum_{y,x \in \mathcal{S}_l} Q_l(y,x)$$

8. TRAINING

It follows from Section 3 that the statistical parameters specifying the SLM are the predictor probabilities $P(v|x,y)$ and the constructor probabilities $Q(a|x,y)$, $v, x, y \in \mathcal{V}$, $a \in \{\mathbf{left}, \mathbf{right}, \mathbf{null}\}$. As usual, we would like to choose these parameters by an appropriate maximum likelihood procedure applied to data. In principle, it would be possible to proceed analogously to the inside – outside algorithm for probabilistic context free grammars [9]. The recursion (10) of Section 5 already corresponds to the inside algorithm and we could develop an outside analogue as well. However, such a re-estimation would be extremely costly.

The simplest way to proceed would be by Viterbi training based on finding the most probable parse $\widehat{\mathbf{T}}$ of the sentence $\mathbf{W}$. Since given any parse $\mathbf{T}$ there is a unique sequence of predictor and constructor actions that achieves it (see Section 3), such re-estimation would simply consist of re-normalization of counts of predictor and constructor actions found in the parses $\widehat{\mathbf{T}}(i)$ that correspond to the sentences $\mathbf{W}(i)$, $i = 1, 2, \ldots, K$ making up the training corpus. Of course, initial statistics would be derived from parses present in some convenient treebank.[10],[11]

For the sake of brevity we now state without proof the basic recursion of the Viterbi algorithm.[8] Let $xy^\dagger[i,j]$ denote the probability, given that x is the last exposed headword and w_i is generated, of the most probable sequence of moves that generate the words $w_{i+1} \ldots, w_j$ with y becoming the headword of the phrase $w_i, w_{i+1} \ldots, w_j$. Then we have for $1 \leq i < j < n+1$ that

$$xy^\dagger[i,j] = \max \left\{ \max_{l \in \{i, j-1\}, z} L(i,j,l,z), \max_{l \in \{i, j-1\}, v} R(i,j,l,v) \right\}$$

where

$$L(i,j,l,z) = xy^\dagger[i,l]\ P^*(w_{l+1}|x,y)\ yz^\dagger[l+1,j]\ Q(\mathbf{left}|y,z)$$
$$R(i,j,l,v) = xv^\dagger[i,l]\ P^*(w_{l+1}|x,v)\ vy^\dagger[l+1,j]\ Q(\mathbf{right}|v,y)$$

with the boundary conditions

$xy^\dagger[i,j] = 0$ if $x \notin \{w_0, \ldots, w_{i-1}\}$ or $y \notin \{w_i, \ldots, w_j\}$ or $i > j$

and

$$xy^\dagger[j,j] = \begin{cases} 1 & \text{for } x \in \{w_0, \ldots, w_{j-1}\}, y = w_j \\ 0 & \text{otherwise} \end{cases}$$

The probability of $\widehat{\mathbf{T}}$ will be given by

$$P(\widehat{\mathbf{T}}, \mathbf{W}) = w_0 w_{n+1}^\dagger[1, n+1]$$

Obviously, the tree $\widehat{\mathbf{T}}$ itself can be obtained by a back-trace of relations (16) starting from the apex of $\widehat{\mathbf{T}}$.

It would be better to base the parameter estimation on more than one parse per training sentence, for instance on the L most probable parses $\widehat{\mathbf{T}}^1, \ldots, \widehat{\mathbf{T}}^L$. In such a case we would weigh the predictor and constructor counter contributions corresponding to the parse $\widehat{\mathbf{T}}^i$ by the probability $P(\widehat{\mathbf{T}}^i, \mathbf{W}) / \sum_{j=1}^L P(\widehat{\mathbf{T}}^j, \mathbf{W})$.

[8] Compare to the development of (9) in Section 5.

The algorithm obtaining the $L-$best parses is computationally quite expensive. An alternative would be to obtain the Viterbi parse $\widehat{\mathbf{T}}^1$ by the recursion (16), and the remaining $L-1$ parses by sampling (with replacement) the parses contained in the chart corresponding to the recursion (10). Such sampling would be carried out top–down. For instance, we see from (10) that a span designated by $xy[i,j]$ is "made up" either of spans $xy[i,l]$ and $yz[l+1,j]$ or of spans $xv[i,l]$ and $vy[l+1,j]$. The sampler would then choose the first span with a probability proportional to $P^*(w_{l+1}|x,y)\,xy[i,l]\;yz[l+1,j]\,Q(\mathbf{left}|y,z)$, etc.

9. SMOOTHING AND PARAMETRIZATION

The basic SLM described in Section 3 involves lexical headwords of phrases that have not been annotated by either non-terminals or parts-of-speech. This presents a problem when estimates of $P(w|h_{-1},h_{-2})$ and $Q(a|h_{-1},h_{-2})$ derived from (necessarily sparse) training data are needed during operation on test data.. The parameter space of these probabilities is just too large. One could try to use standard linear smoothing formulas

$$P(w|h_{-1},h_{-2}) = \lambda_3 f(w|h_{-1},h_{-2}) + \lambda_2 f(w|h_{-1}) + \lambda_1 f(w) \tag{16}$$

and

$$Q(w|h_{-1},h_{-2}) = v_3 f(a|h_{-1},h_{-2}) + v_2 f(a|h_{-1}) + v_1 f(a) \tag{17}$$

to overcome the problem. But formula (17) is particularly problematic: intuition would tell us that the choice of $a \in \{\mathbf{left},\mathbf{right},\mathbf{null}\}$ should depend on both h_{-1} and h_{-2}!

Besides, the partial parse $\mathbf{T}_i$ surely carries a lot of grammatical information that could be taken advantage of. Therefore, the parser should annotate all of its headwords so that in formulas (16) and (17) $h=(v,t)$ where $v \in \mathcal{V}$ (the vocabulary) and $t \in \mathcal{N}$ (the set of non-terminals that includes parts of speech), and $a=(\alpha,t)$ with $\alpha \in \{\mathbf{left},\mathbf{right},\mathbf{null}\}$ and $t \in \mathcal{N}$. This means that in addition to a constructor and predictor, the operation of an SLM must include a *tagger* which tags the just predicted words w_i by parts of speech t with probability $R(t|w,h_{-1},h_{-2})$.

The addition of non-terminal annotation then allows us to replace (17) by the much more sensible (for instance)

$$Q(w|h_{-1},h_{-2}) = v_3 f(a|h_{-1},h_{-2}) + v_2 f(a|t_{-1},t_{-2}) + v_1 f(a)$$

Naturally, the smoothing formula (16) can also be adjusted, at least by

$$P(w|h_{-1},h_{-2}) = \tag{18}$$
$$\lambda_3 f(w|h_{-1},h_{-2}) + \lambda_2 f(w|h_{-1}) + \lambda_4 f(w|v_{-1}) + \lambda_1 f(w)$$

Even with the addition of non-terminal annotation, the proper parametrization of the SLM remains a subject of research. The constructor in particular could benefit from more information about the current partial parse $\mathbf{T}_i$.[9] So h_{-3} might be useful, or at least t_{-3}.

The information extracted from $\mathbf{T}_i$ might be made even more comprehensive if we took advantage of the maximum entropy estimation paradigm [2]. We have had some success with such an approach already [13].

10. PRELIMINARY RESULTS

We have tested the SLM on the Wall Street Journal and Switchboard tasks [5],[12]. Compared to the state-of-the-art trigram language model, the SLM has a lower perplexity by 15% and 5%, respectively. It lowers the recognition error rate (WER) by 1% and 1% absolute, respectively. We are about to carry out experiments on the Broadcast News task.

Because the average sentence length of the Switchboard task is 7 words, the SLM is not really suitable for it.

[9] If the SLM is to remain a language model, the left-to-right development must be strictly adhered to.

REFERENCES

[1] F. Jelinek: *Statistical methods for speech recognition,* MIT Press, Cambridge, MA, 1998

[2] R. Rosenfeld, "A Maximum Entropy Approach to Statistical Language Modeling," *Computer, Speech and Language,* **vol. 10,** 1996

[3] J.R. Bellegarda: "A Latent Semantic Analysis Framework for Large–Span Language Modeling," *Proceedings of Eurospeech 97,* pp. 1451 - 1454, Rhodes, Greece, 1997

[4] C. Chelba and F. Jelinek: "Exploiting Syntactic Structure for Language Modeling," *Proceedings of COLING–ACL,* pp. 225-284, Montreal, CA, 1998

[5] C. Chelba and F. Jelinek: "Recognition Performance of a Structured Language Model," *Proceedings of Eurospeech 99,* to appear, College Park, MD, 1999

[6] J. Cocke, unpublished notes

[7] D.H. Younger: "Recognition and Parsing of Context Free Languages in Time N^3," *Information and Control,* 10, 1967, pp. 198-208

[8] T. Kasami: "An efficient recognition and syntax algorithm for context-free languages," *Scientific Report AFCRL-65-758,* Air Force Cambridge Research Lab., Bedford MA, 1965

[9] J.K. Baker: "Trainable Grammars for Speech Recognition," *Proceedings of the Spring Conference of the Acoustical Society of America,* pp. 547-550, Boston MA, 1979

[10] G. Leech and R. Garside: "Running a Grammar Factory: the Production of Syntactically Analysed Corpora or 'Treebanks'," in: Stig Johansson and Anna-Brita Stenstrom: *English Computer Corpora: Selected Papers and Research Guide*, Mouton de Gruyter, Berlin, 1991.

[11] M. Marcus, B. Santorini, and M. Marcinkiewicz, "Building a large annotated corpus of English: the Penn Treebank," *Computational Linguistics*, **vol. 19**, No. 2, 1993.

[12] C. Chelba and F. Jelinek: "Structured Language Modeling for Speech Recognition," *Proceedings of NLDB99,* to appear, Klagenfurt, Austria, 1999

[13] J. Wu and S. Khudanpur: "Combining Non-local, Syntactic and N-gram dependencies in Language Modeling," *Proceedings of NLDB99,* to appear, Klagenfurt, Austria, 1999