arXiv:cs.CL/0001021 v1 24 Jan 2000

REFINEMENT OF A STRUCTURED LANGUAGE MODEL[†]

Ciprian Chelba, CLSP, The Johns Hopkins University
Frederick Jelinek, CLSP, The Johns Hopkins University

ABSTRACT

A new language model for speech recognition inspired by linguistic analysis is presented. The model develops hidden hierarchical structure incrementally and uses it to extract meaningful information from the word history — thus enabling the use of extended distance dependencies — in an attempt to complement the locality of currently used n-gram Markov models. The model, its probabilistic parametrization, a reestimation algorithm for the model parameters and a set of experiments meant to evaluate its potential for speech recognition are presented.

1 INTRODUCTION

The task of a speech recognizer is to automatically transcribe speech into text. The most successful approach to speech recognition so far is a statistical one [1]: given the observed string of acoustic features A, find the most likely word string $\hat{W}$ among those that could have generated A:

$$\hat{W} = argmax_W P(W|A) = argmax_W P(A|W) \cdot P(W) \tag{1}$$

This paper is concerned with the estimation of the language model probability $P(W)$. We will first describe current modeling approaches to the problem, followed by a detailed explanation of our model. A few preliminary experiments that show the potential of our approach for language modeling will then be presented.

2 BASIC LANGUAGE MODELING

The language modeling problem is to estimate the source probability $P(W)$ where $W = w_1, w_2, \ldots, w_n$ is a sequence of words. This probability is estimated from a text training corpus. Usually the model is parameterized: $P_\theta(W), \theta \in \Theta$ where Θ is referred to as the parameter space. Due to the sequential nature of an efficient search algorithm, the model operates left-to-right, allowing the computation

$$P(w_1, w_2, \ldots, w_n) = P(w_1) \cdot \prod_{i=2}^{n} P(w_i/w_1 \ldots w_{i-1}) \tag{2}$$

[†]This work was funded by the NSF IRI-19618874 grant STIMULATE

We thus seek to develop parametric conditional models:

$$P_\theta(w_i/w_1 \dots w_{i-1}), \theta \in \Theta, w_i \in \mathcal{V} \tag{3}$$

where $\mathcal{V}$ is the vocabulary chosen by the modeler. Currently most successful is the *n-gram language model*:

$$P_\theta(w_i/w_1 \dots w_{i-1}) = P_\theta(w_i/w_{i-n+1} \dots w_{i-1}) \tag{4}$$

2.1 LANGUAGE MODEL QUALITY

All attempts to derive an algorithm that would estimate the model parameters so as to minimize the word error rate have failed. As an alternative, a statistical model is evaluated by how well it predicts a string of symbols W_t — commonly named *test data* — generated by the source to be modeled.

2.1.1 Perplexity

Assume we compare two models M_1 and M_2; they assign probability $P_{M_1}(W_t)$ and $P_{M_2}(W_t)$, respectively, to the sample test string W_t. "Naturally", we consider M_1 to be a better model than M_2 if $P_{M_1}(W_t) > P_{M_2}(W_t)$. The test data is not seen during the model estimation process.

A commonly used quality measure for a given model M is related to the entropy of the underlying source and was introduced under the name of perplexity (PPL) [6]:

$$PPL(M) = exp(-1/|W_t| \sum_{i=1}^{N} \ln [P_M(W_t)]) \tag{5}$$

2.2 SMOOTHING

Assume that our model M is faced with the prediction $w_i|w_1 \dots w_{i-1}$ and that w_i has not been seen in the training corpus in context $w_1 \dots w_{i-1}$ which itself has possibly not been encountered in the training corpus. If $P_M(w_i|w_1 \dots w_{i-1}) = 0$ then $P_M(w_1 \dots w_N) = 0$ thus forcing a recognition error; good models are smooth, in the sense that
$\exists \epsilon(M) > 0$ s.t. $P_M(w_i|w_1 \dots w_{i-1}) > \epsilon, \forall w_i \in \mathcal{V}$, $(w_1 \dots w_{i-1}) \in \mathcal{V}^{i-1}$.

One standard approach that ensures smoothing is the deleted interpolation method [7]. It interpolates linearly among contexts of different order h_n:

$$P_\theta(w_i|w_{i-n+1} \dots w_{i-1}) = \sum_{k=0}^{k=n} \lambda_k \cdot f(w_i/h_k) \tag{6}$$

where: $h_k = w_{i-k+1} \dots w_{i-1}$ is the context of order k when predicting w_i; $f(w_i/h_k)$ is the relative frequency estimate for the conditional probability $P(w_i/h_k)$; $\lambda_k, k = 0 \dots n$ are the interpolation coefficients satisfying $\lambda_k > 0, k = 0 \dots n$ and $\sum_{k=0}^{k=n} \lambda_k = 1$.

The model parameters θ then are: the counts $C(h_n, w_i)$ — lower order counts are inferred recursively by: $C(h_k, w_i) = \sum_{w_{i-k} \in \mathcal{V}} C(w_{i-k}, h_k, w_i)$ — and the interpolation coefficients $\lambda_k, k = 0 \dots n$.

A simple way to estimate the model parameters involves a two stage process:

1. gather counts from *development data* — about 90% of training data;

2. estimate interpolation coefficients to minimize the perplexity of *check data* — the remaining 10% of the training data.

Different smoothing techniques are also used e.g., maximum entropy [2] or back-off [8].

3 DESCRIPTION OF THE STRUCTURED LANGUAGE MODEL

The model we present is closely related to the one investigated in [3], however different in a few important aspects:

- our model operates in a left-to-right manner, thus allowing its use directly in the hypothesis search for $\hat{W}$ in (1);
- our model is a factored version of the one in [3], thus enabling the calculation of the joint probability of words and parse structure; this was not possible in the previous case due to the huge computational complexity of the model.

3.1 THE BASIC IDEA AND TERMINOLOGY

Consider predicting the word `after` in the sentence:
`the contract ended with a loss of 7 cents after trading as low as 89 cents`.
A 3-gram approach would predict `after` from `(7, cents)` whereas it is intuitively clear that the strongest word-pair predictor would be `contract ended` which is outside the reach of even 7-grams. Our assumption is that what enables humans to make a good prediction of `after` is the syntactic structure of its sentence prefix. The linguistically correct *partial parse* of this prefix is shown in Figure 1.

A binary branching parse for a string of words is a binary tree whose leaves are the words. The headword annotation makes the tree an oriented graph: at each node we have two children; the current node receives a *headword* from either child; one arrow suffices to describe which of the children — left or right — is percolated to become the headword of the parent. It was found that better parse trees are generated when using tags: part-of-speech(POS) tags for the leaves and non-terminal(NT) tags for the intermediate nodes in the parse tree. Any subtree identifies a *constituent*. The word `ended` is called the *headword* of the *constituent* `(ended (with (...)))` and `ended` is an *exposed headword* when predicting `after` — topmost headword in the largest constituent that contains it. The syntactic structure in the past filters out irrelevant words and points to the important ones, thus enabling the use of long distance information when predicting the next word.

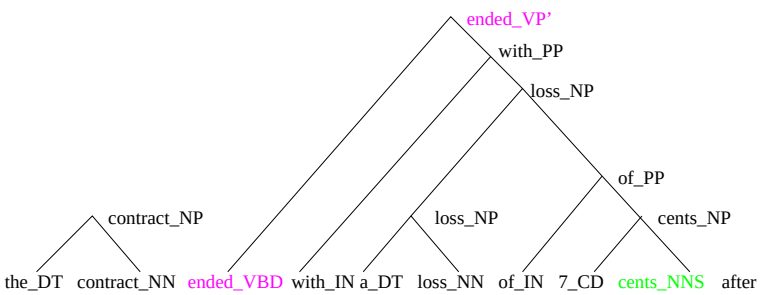

Figure 1: Partial parse

Our model will attempt to build the syntactic structure incrementally while traversing the sentence left-to-right; it will assign a probability $P(W,T)$ to every sentence W with every possible POStag assignment, binary branching parse, non-terminal tag and headword annotation for every constituent of the parse tree T.

Let W be a sentence of length n words to which we have prepended `<s>` and appended `</s>` so that w_0 =`<s>` and w_{n+1} =`</s>`. Let W_k be the word k-prefix $w_0 \ldots w_k$ of the sentence and W_kT_k the *word-parse k-prefix*. A word-parse k-prefix contains — for a given parse — only those binary subtrees whose span is completely included in the word k-prefix, excluding w_0 =`<s>`. Single words along with their POStag can be regarded as root-only subtrees. Figure 2 shows a word-parse k-prefix; `h_0 .. h_{-m}` are the *exposed heads*, each head being a pair (headword, non-terminal tag), or (word, POStag) in the case of a root-only tree.

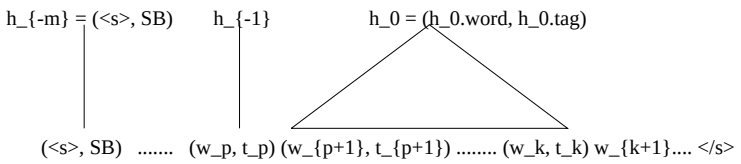

Figure 2: A word-parse k-prefix

A *complete parse* — Figure 3 — is a binary parse of the
`(<s>, SB)` $(w_1,t_1)\ldots(w_n,t_n)$ `(</s>, SE)` sequence with the following two restrictions:

1. $(w_1,t_1)\ldots(w_n,t_n)$ `(</s>, SE)` is a constituent, headed by `(</s>, TOP')`;
2. `(</s>, TOP)` is the only allowed head. Note that $((w_1,t_1)\ldots(w_n,t_n))$ *needn't* be a constituent, but for the parses where it is, there is no restriction on which of its words is the headword or what is the non-terminal tag that accompanies the headword.

Our model can generate all and only the complete parses for a string
`(<s>, SB)` $(w_1,t_1)\ldots(w_n,t_n)$ `(</s>, SE)`.

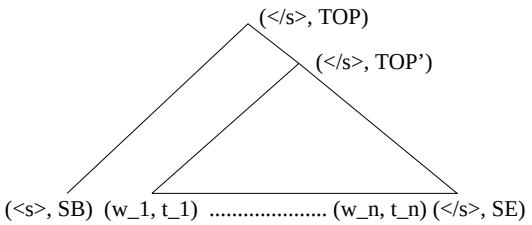

Figure 3: Complete parse

The model will operate by means of three modules:

- WORD-PREDICTOR predicts the next word w_{k+1} given the word-parse k-prefix W_kT_k and then passes control to the TAGGER;
- TAGGER predicts the POStag t_{k+1} of the next word given the word-parse k-prefix and the newly predicted word w_{k+1} and then passes control to the PARSER;

- PARSER grows the already existing binary branching structure by repeatedly generating the transitions: `(adjoin-left, NTtag)` or `(adjoin-right, NTtag)` until it passes control to the PREDICTOR by taking a `null` transition. `NTtag` is the non-terminal tag assigned to each newly built constituent and `{left,right}` specifies from where the new headword is inherited. The parser operates always on the two rightmost exposed heads, starting with the newly tagged word w_{k+1}.

The operations performed by the PARSER are illustrated in Figures 4-6 and they ensure that all possible binary branching parses with all possible headword and non-terminal tag assignments for the $w_1 \ldots w_k$ word sequence can be generated. It is easy

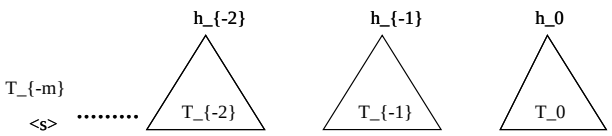

Figure 4: Before an adjoin operation

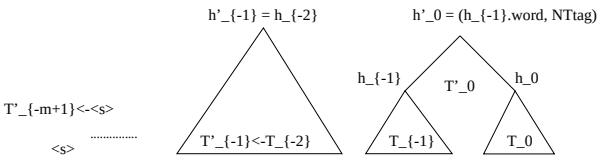

Figure 5: Result of adjoin-left under NTtag

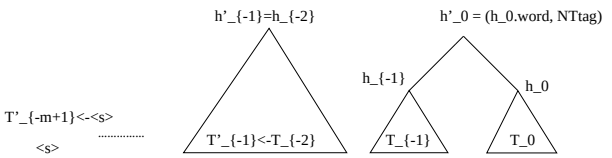

Figure 6: Result of adjoin-right under NTtag

to see that any given word sequence with a possible parse and headword annotation is generated by a unique sequence of model actions.

3.2 PROBABILISTIC MODEL

The probability $P(W,T)$ of a word sequence W and a complete parse T can be broken into:

$$P(W,T) = \prod_{k=1}^{n+1}[P(w_k/W_{k-1}T_{k-1}) \cdot P(t_k/W_{k-1}T_{k-1},w_k) \cdot \prod_{i=1}^{N_k} P(p_i^k/W_{k-1}T_{k-1},w_k,t_k,p_1^k \ldots p_{i-1}^k)] \quad (7)$$

where:

- $W_{k-1}T_{k-1}$ is the word-parse $(k-1)$-prefix
- w_k is the word predicted by WORD-PREDICTOR
- t_k is the tag assigned to w_k by the TAGGER

- $N_k - 1$ is the number of operations the PARSER executes at position k of the input string before passing control to the WORD-PREDICTOR (the N_k-th operation at position k is the `null` transition); N_k is a function of T

- p_i^k denotes the i-th PARSER operation carried out at position k in the word string;
  $p_i^k \in \{$ `(adjoin-left, NTtag), (adjoin-right, NTtag)`$\}, 1 \leq i < N_k$,
  $p_i^k =$`null`$, i = N_k$

Each $(W_{k-1}T_{k-1}, w_k, t_k, p_1^k \ldots p_{i-1}^k)$ is a valid word-parse k-prefix W_kT_k at position k in the sentence, $i = \overline{1, N_k}$.

To ensure a proper probabilistic model certain PARSER and WORD-PREDICTOR probabilities must be given specific values:

- $P($`null`$/W_kT_k) = 1$, if `h_{-1}.word = <s>` and `h_{0}` $\neq$ `(</s>, TOP')` — that is, before predicting `</s>` — ensures that `(<s>, SB)` is adjoined in the last step of the parsing process;

- $P($`(adjoin-right, TOP)`$/W_kT_k) = 1$,
  if `h_0 = (</s>, TOP')` and `h_{-1}.word = <s>` and
  $P($`(adjoin-right, TOP')`$/W_kT_k) = 1$,
  if `h_0 = (</s>, TOP')` and `h_{-1}.word` $\neq$ `<s>` ensure that the parse generated by our model is consistent with the definition of a complete parse;

- $\exists \epsilon > 0, \forall W_{k-1}T_{k-1}, P(w_k=$`</s>`$/W_{k-1}T_{k-1}) \geq \epsilon$ ensures that the model halts with probability one.

In order to be able to estimate the model components we need to make appropriate equivalence classifications of the conditioning part for each component, respectively.

The equivalence classification should identify the strong predictors in the context and allow reliable estimates from a treebank. Our choice is inspired by [4]:

$$P(w_k/W_{k-1}T_{k-1}) = P(w_k/[W_{k-1}T_{k-1}]) = P(w_k/h_0, h_{-1}) \quad (8)$$

$$P(t_k/w_k, W_{k-1}T_{k-1}) = P(t_k/w_k, [W_{k-1}T_{k-1}]) = P(t_k/w_k, h_0.tag, h_{-1}.tag) \quad (9)$$

$$P(p_i^k/W_kT_k) = P(p_i^k/[W_kT_k]) = P(p_i^k/h_0, h_{-1}) \quad (10)$$

It is worth noting that if the binary branching structure developed by the parser were always right-branching and we mapped the POStag and non-terminal tag vocabularies to a single type then our model would be equivalent to a trigram language model.

3.3 SMOOTHING

All model components — WORD-PREDICTOR, TAGGER, PARSER — are conditional probabilistic models of the type $P(y/x_1, x_2, \ldots, x_n)$ where $y, x_1, x_2, \ldots, x_n$ belong to a mixed bag of words, POStags, non-terminal tags and parser operations (y only). For simplicity, the smoothing method we chose was deleted interpolation among relative frequency estimates of different orders $f_n(\cdot)$ using a recursive mixing scheme:

$$P(y/x_1, \ldots, x_n) =$$
$$\lambda(x_1, \ldots, x_n) \cdot P(y/x_1, \ldots, x_{n-1}) + (1 - \lambda(x_1, \ldots, x_n)) \cdot f_n(y/x_1, \ldots, x_n), \quad (11)$$

$$f_{-1}(y) = uniform(vocabulary(y)) \quad (12)$$

The λ coefficients are tied based on the range into which the count $C(x_1, \ldots, x_n)$ falls. The approach is a standard one [7].

3.4 PRUNING STRATEGY

Since the number of parses for a given word prefix W_k grows exponentially with k, $|\{T_k\}| \sim O(2^k)$, the state space of our model is huge even for relatively short sentences. We thus have to prune most parses without discarding the most likely ones for a given sentence W. Our pruning strategy is a synchronous multi-stack search algorithm.

Each stack contains hypotheses — partial parses — that have been constructed by *the same number of predictor and the same number of parser operations*. The hypotheses in each stack are ranked according to the $\ln(P(W_k, T_k))$ score, highest on top. The width of the search is controlled by two parameters:

- the maximum stack depth — the maximum number of hypotheses the stack can contain at any given time;
- log-probability threshold — the difference between the log-probability score of the top-most hypothesis and the bottom-most hypothesis at any given state of the stack cannot be larger than a given threshold.

3.5 WORD LEVEL PERPLEXITY

Attempting to calculate the conditional perplexity by assigning to a whole sentence the probability:

$$P(W/T^*) = \prod_{k=0}^{n} P(w_{k+1}/W_k T_k^*), \tag{13}$$

where $T^* = argmax_T P(W, T)$ — the search for T^* being carried according to our pruning strategy — is not valid because it is not causal: when predicting w_{k+1} we would be using T^* which was determined by looking at the entire sentence. To be able to compare the perplexity of our model with that resulting from the standard trigram approach, we need to factor in the entropy of guessing the prefix of the final best parse T_k^* *before predicting* w_{k+1}, based solely on the word prefix W_k.

To maintain a left-to-right operation of the language model, the probability assignment for the word at position $k+1$ in the input sentence was made using:

$$P(w_{k+1}/W_k) = \sum_{T_k \in S_k} P(w_{k+1}/W_k T_k) \cdot \rho(W_k, T_k), \tag{14}$$

$$\rho(W_k, T_k) = P(W_k T_k) / \sum_{T_k \in S_k} P(W_k T_k) \tag{15}$$

where S_k is the set of all parses present in our stacks at the current stage k.

Note that if we set $\rho(W_k, T_k) = \delta(T_k, T_k^* | W_k)$ — 0-entropy guess for the prefix of the parse T_k to equal that of the final best parse T_k^*— the two probability assignments (13) and (14) would be the same, yielding a lower bound on the perplexity achievable by our model when using a given pruning strategy.

A second important observation is that the next-word predictor probability $P(w_{k+1}/W_k T_k)$ in (14) *need not be the same* as the WORD-PREDICTOR probability (8) used to extract the structure T_k, thus leaving open the possibility to estimate it separately.

3.6 PARAMETER REESTIMATION

3.6.1 First Model Reestimation

Our parameter re-estimation is inspired by the usual EM approach. Let $(W, T^{(k)}), k = 1, 2, \ldots, N$ denote the set of parses of W that survived our pruning strategy. Each parse was produced by a unique sequence of model actions: predictor, tagger, and parser moves. The collection of these moves will be called a *derivation.* Each of the N members of the set is produced by exactly the same number of moves of each type. Each move is uniquely specified by identifiers $(y^{(m)}, \underline{x}^{(m)})$, where $m \in \{$WORD-PREDICTOR, TAGGER, PARSER$\}$ denotes the particular model, $y^{(m)}$ is the specification of the particular move taken (e.g., for m =PARSER, the quantity $y^{(m)}$ specifies a choice from $\{left, right, null\}$ and the exact tag attached), and $\underline{x}^{(m)}$ specifies the move's context (e.g., for m =PARSER, the two heads).

For each possible value $(y^{(m)}, \underline{x}^{(m)})$ we will establish a counter which at the beginning of any particular iteration will be empty. For each move $(y^{(m)}, \underline{x}^{(m)})$ present in the derivation of $(W, T^{(j)})$ we add to the counter specified by $(y^{(m)}, \underline{x}^{(m)})$ the amount

$$\rho(W, T^{(k)}) = \frac{P(W, T^{(k)})}{\sum_{j=1}^{N} P(W, T^{(j)})}$$

where $P(W, T^{(j)})$ are evaluated on the basis of the model's parameter values established at the end of the preceding iteration. We do that for all $(W, T^{(j)}), j = 1, 2, \ldots, N$ and for all sentences W in the training data. Let $C^{(m)}(y^{(m)}, \underline{x}^{(m)})$ be the counter contents at the end of this process. The corresponding relative frequency estimate will be

$$f(y^{(m)} | \underline{x}^{(m)}) = \frac{C^{(m)}(y^{(m)}, \underline{x}^{(m)})}{\sum_{z^{(m)}} C^{(m)}(z^{(m)}, \underline{x}^{(m)})}$$

The lower order frequencies needed for the deleted interpolation of probabilities in the next iteration are derived in the obvious way from the same counters.

It is worth noting that because of pruning (which is a function of the statistical parameters in use), the sets of surviving parses $(W, T^{(k)}), k = 1, 2, \ldots, N$ for the same sentence W may be completely different for different iterations.

3.6.2 First Pass Initial Parameters

Each model component — WORD-PREDICTOR, TAGGER, PARSER — is initialised from a set of hand-parsed sentences, after each parse tree (W, T) is decomposed into its $derivation(W, T)$. Separately for each m model component, we:

- gather joint counts $C^{(m)}(y^{(m)}, \underline{x}^{(m)})$ from the derivations that make up the "development data" using $\rho(W, T) = 1$;
- estimate the deleted interpolation coefficients on joint counts gathered from "check data" using the EM algorithm [5]. These are the initial parameters used with the reestimation procedure described in the previous section.

3.6.3 Language Model Refinement

In order to improve performance, we develop a model to be used in (14), different from the WORD-PREDICTOR model (8). We will call this new component the L2R-WORD-PREDICTOR.

The key step is to recognize in (14) a hidden Markov model (HMM) with fixed transition probabilities — although dependent on the position in the input sentence k — specified by the $\rho(W_k, T_k)$ values.

The Expectation-step of the EM algorithm [5] for gathering joint counts $C^{(m)}(y^{(m)}, \underline{x}^{(m)})$, m = L2R-WORD-PREDICTOR-MODEL, is the standard one whereas the Maximization-step uses the same count smoothing technique as that descibed in section 3.6.1.

The second reestimation pass is seeded with the m = WORD-PREDICTOR model joint counts $C^{(m)}(y^{(m)}, \underline{x}^{(m)})$ resulting from the first parameter reestimation pass (see section 3.6.1).

4 EXPERIMENTS

We have carried out the reestimation technique described in section 3.6 on 1 Mwds of "development" data. For convenience we chose to work on the UPenn Treebank corpus [9] — a subset of the WSJ (Wall Stree Journal) corpus. The vocabulary sizes were:
word vocabulary: 10k, open — all words outside the vocabulary are mapped to the `<unk>` token; POS tag vocabulary: 40, closed; non-terminal tag vocabulary: 52, closed; parser operation vocabulary: 107, closed. The development set size was 929,564wds (sections 00-20), check set size 73,760wds (sections 21-22), test set size 82,430wds (sections 23-24).

Table 1 shows the results of the reestimation techniques presented in section 3.6; `E?` and `L2R?` denote iterations of the reestimation procedure described in sections 3.6.1 and 3.6.3, respectively. A deleted interpolation trigram model had perplexity 167.14 on the same training-test data.

iteration number	DEV set L2R-PPL	TEST set L2R-PPL
E0	24.70	167.47
E1	22.34	160.76
E2	21.69	158.97
E3 = L2R0	21.26	158.28
L2R5	17.44	153.76

Table 1: Parameter reestimation results

Simple linear interpolation between our model and the trigram model:

$$Q(w_{k+1}/W_k) = \lambda \cdot P(w_{k+1}/w_{k-1}, w_k) + (1 - \lambda) \cdot P(w_{k+1}/W_k)$$

yielded a further improvement in PPL, as shown in Table 2. The interpolation weight was estimated on check data to be $\lambda = 0.36$. An overall relative reduction of 11% over the trigram model has been achieved.

As outlined in section 3.5, the perplexity value calculated using (13) is a lower bound for the achievable perplexity of our model; for the above search parameters and E3 model statistics this bound was 99.60, corresponding to a relative reduction of 41% over the trigram model.

iteration number	TEST set L2R-PPL	TEST set 3-gram interpolated PPL
E0	167.47	152.25
E3	158.28	148.90
L2R5	153.76	147.70

Table 2: Interpolation with trigram results

5 CONCLUSIONS AND FUTURE DIRECTIONS

A new source model that organizes the prefix hierarchically in order to predict the next symbol is developed. As a case study we applied the source model to natural language, thus developing a new language model with applicability in speech recognition.

We believe that the above experiments show the potential of our approach for improved language modeling for speech recognition. Our future plans include:

- experiment with other parameterizations for the word predictor and parser models;
- evaluate model performance as part of an automatic speech recognizer (measure word error rate improvement).

6 Acknowledgments

This research has been funded by the NSF IRI-19618874 grant (STIMULATE).

The authors would like to thank to Sanjeev Khudanpur for his insightful suggestions. Also to Harry Printz, Eric Ristad, Andreas Stolcke, Dekai Wu and all the other members of the dependency modeling group at the summer96 DoD Workshop for useful comments on the model, programming support and an extremely creative environment. Also thanks to Eric Brill, Sanjeev Khudanpur, David Yarowsky, Radu Florian, Lidia Mangu and Jun Wu for useful input during the meetings of the people working on our STIMULATE grant.

REFERENCES

[1] L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume PAMI-5, pages 179–90, March 1983.

[2] A. L. Berger, S. A. Della Pietra, and V. J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–72, March 1996.

[3] C. Chelba, D. Engle, F. Jelinek, V. Jimenez, S. Khudanpur, L. Mangu, H. Printz, E. S. Ristad, R. Rosenfeld, A. Stolcke, and D. Wu. Structure and performance of a dependency language model. In *Proceedings of Eurospeech*, volume 5, pages 2775–2778. Rhodes, Greece, 1997.

[4] M. J. Collins. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 184–191. Santa Cruz, CA, 1996.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. In *Journal of the Royal Statistical Society*, volume 39 of *B*, pages 1–38. 1977.

[6] F. Jelinek. *Information Extraction From Speech And Text.* MIT Press, 1997.

[7] F. Jelinek and R. Mercer. Interpolated estimation of markov source parameters from sparse data. In E. Gelsema and L. Kanal, editors, *Pattern Recognition in Practice*, pages 381–397. 1980.

[8] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, volume 35, pages 400–01, March 1987.

[9] M. Marcus, B. Santorini, and M. Marcinkiewicz. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1995.