Neural Probabilistic Structured Language Modeling

Ahmad Emami and Frederick Jelinek
CLSP/Electrical and Computer Engineering

AIMS We investigate the performance of the Structured Language Model when one of its components is modeled by a connectionist model. Using a connectionist model and a distributed representation of the items in the history allows the component to use much longer contexts than possible with current interpolated or back-off models, both because of the inherent capability of the connectionist model to fight the data sparseness problem, and because of the only sub-linear growth in the model size when increasing the context length.

INTRODUCTION A language model is a main component of many systems dealing with speech or natural language such as speech recognition or machine translation systems. N-gram language models are used in all of the current practical state-of-the-art systems. In these models only the surface (words only) information, and that only limited to a short span (last N-1 words), is used to predict the next word and the prediction is based on how often a given N-gram was seen in the training data. There have been attempts to use longer contexts by means of inducing some manageable number of features from the whole past and using them for the prediction. In one such method, the Structured Language Model (SLM) [1], partial syntactical parses are built on the past sequence of words and a subset of information (features) gained from the parses is used to predict the next word. By using only a small number of features obtained from a parse of a long past, the Structured Language Model avoids the data sparseness problem without limiting itself to a short context.

There has been recent promising work in using distributional representation of words and neural networks for Language Modeling [2]. One great advantage of this approach is its ability to fight data sparseness. The model size grows only sub-linearly with the number of predicting features used. It has been shown that this method improves on regular N-gram models in both perplexity and word error rate [2, 3]. In this work we investigate the impact of using a neural network model as the predictor component of the Structured Language Model, giving it the ability to use many more features than the original SLM while avoiding data sparseness.

STRUCTURED LANGUAGE MODEL An extensive presentation of the SLM can be found in [1]. The model assigns a probability $P(W,T)$ to every sentence W and every possible binary parse T of W. The terminals of T are the words of W with POS tags, and the nodes of T are annotated with phrase headwords and non-terminal labels.

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

$$\begin{aligned} P(W,T) = & \\ & \prod_{k=1}^{n+1}[P(w_k|W_{k-1}T_{k-1})\cdot P(t_k|W_{k-1}T_{k-1},w_k)\cdot \\ & \prod_{i=1}^{N_k} P(p_i^k|W_{k-1}T_{k-1},w_k,t_k,p_1^k...p_{i-1}^k)] \end{aligned} \quad (1)$$

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 CONSTRUCTOR executes at sentence position k 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 $i - th$ CONSTRUCTOR operation carried out at position k in the word string; the operations performed by the CONSTRUCTOR ensure that all possible binary branching parses, with all possible headword and non-terminal label assignments for the $w_1 \ldots w_k$ word sequence, can be generated. The $p_1^k \ldots p_{N_k}^k$ sequence of CONSTRUCTOR operations at position k grows the word-parse $(k-1)$-prefix into a word-parse k-prefix.

The *language model* probability assignment for the word at position $k+1$ in the input sentence is made using:

$$P_{SLM}(w_{k+1}|W_k) = \sum_{T_k \in S_k} P(w_{k+1}|W_k T_k) \cdot \rho(W_k, T_k), \quad (2)$$

$$\rho(W_k, T_k) = P(W_k T_k) / \sum_{T_k \in S_k} P(W_k T_k), \quad (3)$$

which ensures a proper probability normalization over strings W^*, where S_k is the set of all parses present in our stacks at the current stage k.

NEURAL NETWORK MODEL In brief, this model can be described as follows: a *feature vector* is associated with each token in some given *input vocabulary.* The input to the neural network is then composed of a single vector which is a concatenation of all the feature vectors of the items we are conditioning upon. The neural network itself computes the (conditional) distribution over all the tokens in the given *output vocabulary* given the input described above. The conditional probability function $P(y|x_1, x_2, \cdots, x_{n-1})$ where x_i and y are from the input and output vocabularies V_i and V_o respectively, is determined in two parts:

1. A mapping that associates with each word in the input vocabulary V_i a real vector of fixed length

2. A conditional probability function which takes as the input the concatenation of the feature vectors of the input items $x_1, x_2, \cdots, x_{n-1}$. The function produces a probability distribution (a vector) over V_o, the $i - th$ element being the conditional probability of the $i - th$ member of V_o. This probability function is realized by a standard multi-layer neural network. A *softmax* function is used at the output of the neural net to make sure probabilities sum up to 1.

Training is achieved by searching for parameters Φ of the neural network and the values of feature vectors that maximize the penalized log-likelihood of the training corpus:

$$L = \frac{1}{T} \sum_t log P(y^t | x_1^t, \ldots, x_{n-1}^t; \Phi) + R(\Phi) \quad (4)$$

where $P(y^t|x_1^t, ..., x_{n-1}^t)$ is the probability of word y^t (network output at time t), T is the training data size and $R(\Phi)$ is a regularization term, sum of the parameters' squares in our case.

NEURAL NETWORK IN SLM In this work we investigate the use of a neural net model as the PREDICTOR component of the Structured Language Model. The other components of the SLM remain unchanged. The reason that only PREDICTOR was chosen for neural network implementation was that previous experiments with the SLM have shown that the data sparseness problem is much more severe for the PREDICTOR than for the other components and in fact the

		+slm	+3gm	+5gm
SLM	161	161	137	132
2HW	174	137	127	123
3HW	161	132	123	119
HW-OP	155	129	121	**117**

Table 1: UPENN section perplexity

perplexity of both the TAGGER and the CONSTRUCTOR were found to be very close to 1.

The baseline Structured Language Model uses different conditioning contexts for its different components. The PREDICTOR uses the two previous heads as the context. The CONSTRUCTOR uses the same context plus the non-terminal tag of the third previous headword, and finally the TAGGER uses the current word and tags of the two previous heads as its context.

Table 1 gives the perplexity results on the UPENN section of the Wall Street Journal (WSJ) corpus. The vocabulary size was 10,000 and there were a total of 94 non-terminal tags and part of speech tags. The rows denoted by 2HW, 3HW, and HW-OP correspond to contexts consisting of 2 previous heads, 3 previous heads, and 3 previous heads plus the first previous opposite head. The $n-th$ previous opposite head is the child of the $n-th$ previous head that is not the head itself. The columns +slm, +3gm, and +5gm denote linear interpolation with the baseline SLM, a 3-gram back-off, and 5-gram back-off models respectively, with weights found on some held-out set. It can be seen that adding more context always helps and the best result on the longest context improves on the best result for the baseline significantly. The interesting point is that the neural network model seems to be much more uncorrelated with the 3-gram or 5-gram models than is the standard SLM. This can be observed best for the shortest context where the connectionist model performs worse than the SLM baseline, but then does significantly better when interpolated with a regular N-gram model. The SLM model simply can't reproduce the same improvements when combined with the same N-gram models.

With significant improvements gained on the perplexity we carried out some experiments to see how well the model performs its function in speech recognition and in reducing the word error rate. Our model was used to re-rank the output N-best list of a speech recognizer on the Wall Street Journal Corpus. The vocabulary was 19,006 and there were again 94 non-terminal tag and part of speech types. Because of the larger size of the WSJ corpus the training time was considerably longer than for the UPENN corpus. We limited the size of the output vocabulary to 5000 words, reducing computations almost proportionally to the reduction in vocabulary size [3]. The out of vocabulary (OOV) rate for this limited vocabulary is rather low so we were not expecting a major degradation in performance. For the words outside this limited vocabulary we used the regular back-off 5-gram probabilities. Substituting probabilities in this manner causes them not to sum up to one anymore but this is not critical since we are using the probabilities only as scores to re-rank the recognizer's hypotheses. The rest of the network is the same as in the perplexity experiment except that the number of training iterations was limited to a maximum of 30. We should also note that the neural network was trained only on half of the available WSJ data, the same amount the baseline SLM is trained on. That is also the case for the 3-gram and 5-gram back-off models we used but the language model of the recognizer is trained on the whole WSJ corpus. The results are given in Table 2. Here the row Lattice denotes the language model from the speech recognizer and the column +l&5gm denotes interpolation with the lattice and back-off 5-gram models. The

		+slm	+lattice	+5gm	+l&5gm
Lattice	13.7	12.6	13.7	13.2	13.2
SLM	12.7	12.7	12.6	12.7	12.6
2HW	13.5	12.7	12.7	12.5	**12.4**
3HW	13.6	12.6	12.8	12.7	12.6
HW-OP	13.2	12.5	12.9	**12.4**	**12.4**

Table 2: WSJ word error rate

linear interpolation weights were chosen to give the best performance on the test set itself.

The results on word error rate don't constitute as good an improvement as we obtained for perplexity and we think that the main reason is that the model is optimized explicitly for perplexity without any consideration for its performance on word error rate reduction.

FUTURE WORK In this work the neural network model used as the PREDICTOR of the SLM was trained on the single Gold Standard parses (produced either by humans or an reliable outside parser), however the evaluation was on the N-best parses produced by the standard SLM. It would be interesting to train the neural network model on actually the N-best parses produced by the SLM.

We also plan to extend the use of the neural network to all the components of the SLM. We then can perform the whole training of the SLM using neural networks, starting with the Gold Standard parses, training the neural network components, letting the SLM produce N-best parses using the trained components, re-training the components on the produced N-best parses, and again producing N-best parses. The last 2 steps will be iterated until convergence is achieved.

References

[1] Ciprian Chelba and Frederick Jelinek, "Structured language modeling," *Computer Speech and Language*, vol. 14, no. 4, pp. 283–332, October 2000.

[2] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," in *Advances in Neural Information Processing Systems*, 2001.

[3] Holger Schwenk and Jean-Luc Gauvain, "Connectionist language modeling for large vocabulary continuous speech recognition," in *Proc. ICASSP*, 2002.

[4] Simon Haykin, *Neural Networks, A Comprehensive Foundation*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.