Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 191-198.

A Study on Richer Syntactic Dependencies for Structured Language Modeling

Peng Xu
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218
xp@clsp.jhu.edu

Ciprian Chelba
Microsoft Research
One Microsoft Way
Redmond, WA 98052
chelba@microsoft.com

Frederick Jelinek
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218
jelinek@clsp.jhu.edu

Abstract

We study the impact of richer syntactic dependencies on the performance of the structured language model (SLM) along three dimensions: parsing accuracy (LP/LR), perplexity (PPL) and word-error-rate (WER, N-best re-scoring). We show that our models achieve an improvement in LP/LR, PPL and/or WER over the reported baseline results using the SLM on the UPenn Treebank and Wall Street Journal (WSJ) corpora, respectively. Analysis of parsing performance shows correlation between the quality of the parser (as measured by precision/recall) and the language model performance (PPL and WER). A remarkable fact is that the enriched SLM outperforms the baseline 3-gram model in terms of WER by 10% when used in isolation as a second pass (N-best re-scoring) language model.

1 Introduction

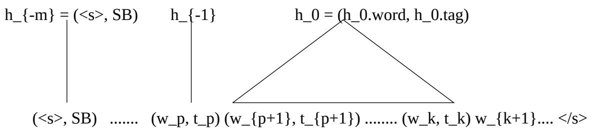
The structured language model uses hidden parse trees to assign conditional word-level language model probabilities. As explained in (Chelba and Jelinek, 2000), Section 4.4.1, if the final best parse is used to be the only parse, the reduction in PPL —relative to a 3-gram baseline— using the SLM's headword parametrization for word prediction is about 40%. The key to achieving this reduction is a good guess of the final best parse for a given sentence as it is being traversed left-to-right, which is much harder than finding the final best parse for the entire sentence, as it is sought by a regular statistical parser. Nevertheless, it is expected that techniques developed in the statistical parsing community that aim at recovering the best parse for an entire sentence, i.e. as judged by a human annotator, should also be productive in enhancing the performance of a language model that uses syntactic structure.

The statistical parsing community has used various ways of enriching the dependency structure underlying the parametrization of the probabilistic model used for scoring a given parse tree (Charniak, 2000) (Collins, 1999). Recently, such models (Charniak, 2001) (Roark, 2001) have been shown to outperform the SLM in terms of both PPL and WER on the UPenn Treebank and WSJ corpora, respectively. In (Chelba and Xu, 2001), a simple way of enriching the probabilistic dependencies in the CONSTRUCTOR component of the SLM also showed better PPL and WER performance; the simple modification to the training procedure brought the WER performance of the SLM to the same level with the best as reported in (Roark, 2001).

In this paper, we present three simple ways of enriching the syntactic dependency structure in the SLM, extending the work in (Chelba and Xu, 2001). The results show that an improved parser (as measured by LP/LR) is indeed helpful in reducing the PPL and WER. Another remarkable fact is that for the first time a language model exploiting elementary syntactic dependencies obviates the need for interpolation with a 3-gram model in N-best rescoring.

2 SLM Review

An extensive presentation of the SLM can be found in (Chelba and Jelinek, 2000). The model assigns a probability $P(W,T)$ to every sentence W and every possible binary parse T. 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. Let W be a sentence of length n



h_{-m} = (<s>, SB)
h_{-1}
h_0 = (h_0.word, h_0.tag)
(<s>, SB) .... (w_p, t_p) (w_{p+1}, t_{p+1}) .... (w_k, t_k) w_{k+1}.... </s>

Figure 1: A word-parse k-prefix

words to which we have prepended the sentence beginning marker `<s>` and appended the sentence end marker `</s>` so that w_0 =`<s>` and w_{n+1} =`</s>`. Let $W_k = w_0 \ldots w_k$ be the word k-prefix of the sentence — the words from the beginning of the sentence up to the current position k — and $W_k T_k$ the *word-parse k-prefix*. Figure 1 shows a word-parse k-prefix; `h_0, .., h_{-m}` are the *exposed heads*, each head being a pair (headword, non-terminal label), or (word, POS tag) in the case of a root-only tree. The exposed heads at a given position k in the input sentence are a function of the word-parse k-prefix.

2.1 Probabilistic Model

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

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

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 are illustrated in Figures 2-3 and they 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.

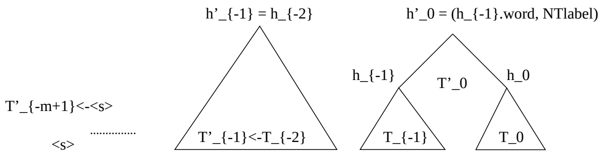


Figure 2: Result of adjoin-left under NT label

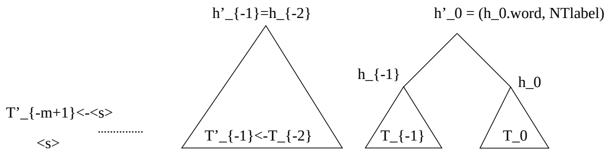


Figure 3: Result of adjoin-right under NT label

The SLM is based on three probabilities, each estimated using deleted interpolation and parameterized (approximated) as follows:

$$P(w_k/W_{k-1}T_{k-1}) = P(w_k/h_0,h_{-1}), \quad (2)$$
$$P(t_k/w_k,W_{k-1}T_{k-1}) = P(t_k/w_k,h_0,h_{-1}), \quad (3)$$
$$P(p_i^k/W_kT_k) = P(p_i^k/h_0,h_{-1}). \quad (4)$$

It is worth noting that if the binary branching structure developed by the parser were always right-branching and we mapped the POS tag and non-terminal label vocabularies to a single type, then our model would be equivalent to a trigram language model. 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, so we have to use a search strategy that prunes it. One choice is a synchronous multi-stack search algorithm which is very similar to a beam search.

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

$$\begin{aligned} P_{SLM}(w_{k+1}/W_k) &= \textstyle\sum_{T_k \in S_k} P(w_{k+1}/W_k T_k) \cdot \rho(W_k, T_k), \\ \rho(W_k, T_k) &= P(W_k T_k) / \textstyle\sum_{T_k \in S_k} P(W_k T_k), \end{aligned} \quad (5)$$

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.

Each model component —WORD-PREDICTOR, TAGGER, CONSTRUCTOR— is initialized from a set of parsed sentences after undergoing headword percolation and binarization, see Section 2.2. An N-best EM (Dempster et al., 1977) variant is then employed to jointly reestimate the model parameters such that the PPL on training data is decreased — the likelihood of the training data under our model is increased. The reduction in PPL is shown experimentally to carry over to the test data.

2.2 Headword Percolation And Binarization

As explained in the previous section, the SLM is initialized on parse trees that have been binarized and the non-terminal (NT) tags at each node have been enriched with headwords. We will briefly review the headword percolation and binarization procedures; they are explained in detail in (Chelba and Jelinek, 2000).

The position of the headword within a constituent — equivalent to a context-free production of the type $Z \to Y_1 \ldots Y_n$, where $Z, Y_1, \ldots Y_n$ are NT labels or POS tags (only for Y_i) — is specified using a rule-based approach.

Assuming that the index of the headword on the right-hand side of the rule is k, we binarize the constituent as follows: depending on the Z identity we apply one of the two binarization schemes in Figure 4. The intermediate nodes created by the above binarization schemes receive the NT label Z'[1]. The choice among the two schemes is made according to a list of rules based on the identity of the label on the left-hand-side of a CF rewrite rule.

3 Enriching Syntactic Dependencies

The SLM is a strict left-to-right, bottom-up parser, therefore in Eq.(2, 3, 4) the probabilities are conditioned on the left contextual information. There are two main reasons we prefer strict left-to-right parsers for the purpose of language modeling (Roark, 2001):

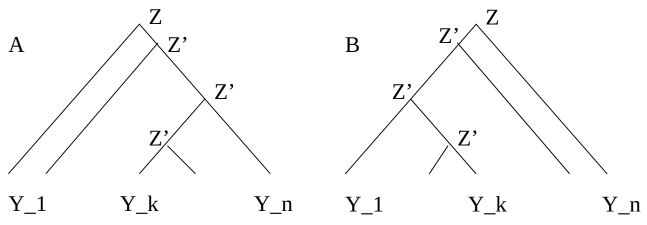


Figure 4: Binarization schemes

- when looking for the most likely word string given the acoustic signal (as required in a speech recognizer), the search space is organized as a prefix tree. A language model whose aim is to guide the search must thus operate left-to-right.
- previous results (Chelba and Jelinek, 2000) (Charniak, 2001) (Roark, 2001) show that a grammar-based language model benefits from interpolation with a 3-gram model. Strict left-to-right parsing makes it easy to combine with a standard 3-gram at the word level (Chelba and Jelinek, 2000) (Roark, 2001) rather than at sentence level (Charniak, 2001).

For these reasons, we prefer enriching the syntactic dependencies by information from the left context.

However, as mentioned in (Roark, 2001), one way of conditioning the probabilities is by annotating the extra conditioning information onto the node labels in the parse tree. We can annotate the training corpus with richer information and with the same SLM training procedure we can estimate the probabilities under the richer syntactic tags. Since the treebank parses allow us to annotate parent information onto the constituents, as Johnson did in (Johnson, 1998), this richer predictive annotation can extend information slightly beyond the left context.

Under the equivalence classification in Eq.(2, 3, 4), the conditional information available to the SLM model components is made up of the two most-recent exposed heads consisting of two NT tags and two headwords. In an attempt to

[1]Any resemblance to X-bar theory is purely coincidental.

extend the syntactic dependencies beyond this level, we enrich the non-terminal tag of a node in the binarized parse tree with the NT tag of the parent node, or the NT tag of the child node from which the headword is not being percolated (same as in (Chelba and Xu, 2001)), or we add the NT tag of the third most-recent exposed head to the history of the CONSTRUCTOR component. The three ways are briefly described as:

1. opposite (OP): we use the non-terminal tag of the child node from which the headword is not being percolated
2. parent (PA): we use the non-terminal tag of the parent node to enrich the current node
3. h-2: we enrich the conditioning information of the CONSTRUCTOR with the non-terminal tag of the third most-recent exposed head, but not the headword itself. Consequently, Eq. 4 becomes

$$P(p_i^k/W_kT_k) = P(p_i^k/h_0, h_{-1}, h_{-2}.tag)$$

We take the example from (Chelba and Xu, 2001) to illustrate our enrichment approaches. Assume that after binarization and headword percolation, we have a noun phrase constituent:

```
(NP_group
    (DT the)
    (NP'_group
        (NNP dutch)
        (NP'_group (VBG publishing)
                   (NN group)))),
```

which, after enriching the non-terminal tags using the *opposite* and *parent* scheme, respectively, becomes

```
(NP+DT_group
    (DT the)
    (NP'+NNP_group
         (NNP dutch)
         (NP'+VBG_group (VBG publishing)
                        (NN group))))
```

and[2]

```
(NP+*_group
    (DT+NP the)
    (NP'+NP_group
         (NNP+NP' dutch)
         (NP'+NP'_group (VBG+NP' publishing)
                        (NN+NP' group)))).
```

[2]The NP+* has not been enriched yet because we have not specifi ed the NT tag of the parent of the NP_group

A given binarized tree is traversed recursively in depth-first order and each constituent is enriched in the *parent* or *opposite* manner or both. Then from the resulting parse trees, all three components of the SLM are initialized and N-best EM training can be started.

Notice that both *parent* and *opposite* affect all three components of the SLM since they change the NT/POS vocabularies, but *h-2* only affects the CONSTRUCTOR component. So we believe that if *h-2* helps in reducing PPL and WER, it's because we have thereby obtained a better parser. We should also notice the difference between *parent* and *opposite* in the bottom-up parser. In *opposite* scheme, POS (part of speech) tags are not enriched. As we parse the sentence, two most-recent exposed heads will be adjoined together under some enriched NT label (Figure 2, 3), the NT label has to match the NT tag of the child node from which the headword is not being percolated. Since the NT tags of the children are already known at the moment, the *opposite* scheme actually restricts the possible NT labels. In the *parent* scheme, POS tags are also enriched with the NT tag of the parent node. When a POS tag is predicted from the TAGGER, actually both the POS tag and the NT tag of the parent node are hypothesized. Then when two most recent exposed heads are adjoined together under some enriched NT label, the NT label has to match the parent NT information carried in both of the exposed heads. In other words, if the two exposed heads bear different information about their parents, they can never be adjoined. Since this restriction of adjoin movement is very tight, pruning may delete some or all the good parsing hypotheses early and the net result may be later development of inadequate parses which lead to poor language modeling and poor parsing performance.

Since the SLM parses sentences bottom-up while the parsers used in (Charniak, 2000), (Charniak, 2001) and (Roark, 2001) are top-down, it's not clear how to find a direct correspondence between our schemes of enriching the dependency structure and the ones employed above. However, it is their "pick-and-choose" strategy that inspired our study of richer syntactic dependencies for the SLM.

Model	Word	NT	POS	Parser
baseline & h-2	10001	54	40	163
PA & h-2+PA	10001	570	620	1711
OP & h-2+OP	10001	970	40	2863
OP+PA & h-2+OP+PA	10001	3906	620	11719

Table 1: Vocabulary size comparison of the models

4 Experiments

With the three enrichment schemes described in Section 3 and their combinations, we evaluated the PPL performance of the resulting seven models on the UPenn Treebank and the WER performance on the WSJ setup, respectively. In order to see the correspondence between parsing accuracy and PPL/WER performance, we also evaluated the labeled precision and recall statistics (LP/LR, the standard parsing accuracy measures) on the UPenn Treebank corpus. For every model component in our experiments, deleted-interpolation was used for smoothing. The interpolation weights were estimated from separate held-out data. For example, in the UPenn Treebank setup, we used section 00-20 as training data, section 21-22 as held-out data, and section 23-24 as test data.

4.1 Perplexity

We have evaluated the perplexity of the seven different models, resulting from applying *parent*, *opposite*, *h-2* and their combinations. For each way of initializing the SLM we have performed 3 iterations of N-best EM training. The SLM is interpolated with a 3-gram model, built from exactly the same training data and word vocabulary, using a fixed interpolation weight. As we mentioned in Section 3, the NT/POS vocabularies for the seven models are different because of the enrichment of NT/POS tags. Table 1 shows the actual vocabulary size we used for each model (for parser, the vocabulary is a list of all possible parser operations). The *baseline* model is the standard SLM as described in (Chelba and Jelinek, 2000).

The PPL results are summarized in Table 2. The SLM is interpolated with a 3-gram model as shown in the equation:

$$P(\cdot) = \lambda \cdot P_{3-gram}(\cdot) + (1-\lambda) \cdot P_{SLM}(\cdot).$$

Model	Iter	λ=0.0	λ=0.4
baseline	0	167.4	151.9
baseline	3	158.7	148.7
PA	0	187.6	154.5
PA	3	164.5	149.5
OP	0	157.9	147.0
OP	3	151.2	144.2
OP+PA	0	185.2	152.1
OP+PA	3	162.2	147.3
h-2	0	161.4	149.2
h-2	3	159.4	148.2
h-2+PA	0	163.7	144.7
h-2+PA	3	160.5	143.9
h-2+OP	0	154.8	145.1
h-2+OP	3	153.6	144.4
h-2+OP+PA	0	165.7	144.1
h-2+OP+PA	3	165.4	143.8

Table 2: SLM PPL results

Model	Iter=0		Iter=3	
	LP	LR	LP	LR
baseline	69.22	61.56	69.01	57.82
PA	79.84	45.46	81.20	39.52
OP	74.55	62.97	72.54	59.76
OP+PA	82.58	45.57	83.62	39.54
h-2	73.72	72.27	73.24	71.13
h-2+PA	75.59	70.93	74.93	70.56
h-2+OP	76.91	73.89	76.11	72.65
h-2+OP+PA	78.35	66.04	77.73	64.95

Table 3: Labeled precision/recall(%) results

We should note that the PPL result of the 3-gram model is 166.6. As we can see from the table, without interpolating with the 3-gram, the *opposite* scheme performed the best, reducing the PPL of the *baseline* SLM by almost 5% relative. When the SLM is interpolated with the 3-gram, the *h-2+opposite+parent* scheme performed the best, reducing the PPL of the *baseline* SLM by 3.3%. However, the *parent* and *opposite+parent* schemes are both worse than the *baseline*, especially before the EM training and with λ=0.0. We will discuss the results further in Section 4.4.

4.2 Parsing Accuracy Evaluation

Table 3 shows the labeled precision/recall accuracy results. The labeled precision/recall results of our model are much worse than those reported in (Charniak, 2001) and (Roark, 2001). One of the reasons is that the SLM was not aimed at being a parser, but rather a language model. Therefore, in the search algorithm, the end-of-sentence symbol

Model	Iter	λ^*	λ					
			0.0	0.2	0.4	0.6	0.8	1.0
baseline	0	13.0	13.1	13.1	13.1	**13.0**	13.4	13.7
PA	0	13.0	13.1	13.1	12.9	12.9	13.1	13.7
OP	0	12.8	12.7	12.8	12.8	12.7	13.1	13.7
OP+PA	0	13.1	13.3	12.9	13.0	12.9	13.1	13.7
h-2	0	12.5	12.7	12.5	12.6	12.9	13.2	13.7
h-2+ PA	0	12.7	12.8	13.0	12.7	12.7	13.0	13.7
h-2+ OP	0	<u>12.3</u>	12.3	12.4	12.6	12.7	12.8	13.7
h-2+ OP+ PA	0	12.6	12.6	12.4	12.5	12.7	12.9	13.7

Table 4: N-best re-scoring WER(%) results

can be predicted before the parse of the sentence is ready for completion[3], thus completing the parse with a series of special CONSTRUCTOR moves (see (Chelba and Jelinek, 2000) for details). The SLM allows right-branching parses which are not seen in the UPenn Treebank corpus and thus the evaluation against the UPenn Treebank is inherently biased.

It can also be seen that both the LP and the LR dropped after 3 training iterations: the N-best EM variant used for SLM training algorithm increases the likelihood of the training data, but it cannot guarantee an increase in LP/LR, since the re-estimation algorithm does not explicitly use parsing accuracy as a criterion.

4.3 N-best Re-scoring Results

To test our enrichment schemes in the context of speech recognition, we evaluated the seven models in the WSJ DARPA'93 HUB1 test setup. The same setup was also used in (Roark, 2001), (Chelba and Jelinek, 2000) and (Chelba and Xu, 2001). The size of the test set is 213 utterances, 3446 words. The 20k words open vocabulary and baseline 3-gram model are the standard ones provided by NIST and LDC — see (Chelba and Jelinek, 2000) for details. The lattices and N-best lists were generated using the standard 3-gram model trained on 45M words of WSJ. The N-best size was at most 50 for each utterance, and the average size was about 23. The SLM was trained on 20M words of WSJ text automatically parsed using the parser in (Ratnaparkhi, 1997), binarized and enriched with headwords and NT/POS tag information as explained in Section 2.2 and Section 3. Because SLM training on the 20M words of WSJ text is very expensive, especially after enriching the NT/POS tags, we only evaluated the WER performance of the seven models with initial statistics from binarized and enriched parse trees. The results are shown in Table 4. The table shows not only the results according to different interpolation weights λ, but also the results corresponding to λ^*, a virtual interpolation weight. We split the test data into two parts, A and B. The best interpolation weight, estimated from part A, was used to decode part B, and vice versa. We finally put the decoding results of the two parts together to get the final decoding output. The interpolation weight λ^* is virtual because the best interpolation weights for the two parts might be different. Ideally, λ should be estimated from separate held-out data and then applied to the test data. However, since we have a small number of N-best lists, our approach should be a good estimate of the WER under the ideal interpolation weight.

As can be seen, the *h-2+opposite* scheme achieved the best WER result, with a 0.5% absolute reduction over the performance of the *opposite* scheme. Overall, the enriched SLM achieves 10% relative reduction in WER over the 3-gram model baseline result($\lambda = 1.0$).

The SLM enriched with the *h-2+opposite* scheme outperformed the 3-gram used to generate the lattices and N-best lists, without interpolating it with the 3-gram model. Although the N-best lists are already highly restricted by the 3-gram model during the first recognition pass, this fact still shows the potential of a good grammar-based language model. In particular, we should notice that the SLM was trained on 20M words of WSJ while the lattice 3-gram was trained on 45M words of WSJ. However, our results are not indicative of the performance of SLM as a first pass language model.

4.4 Discussion

By enriching the syntactic dependencies, we expect the resulting models to be more accurate and thus

[3] A parse is ready for completion when at the end of the sentence there are exactly two exposed headwords, the first of which if the start-of-sentence symbol and the second is an ordinary word. See (Chelba and Jelinek, 2000) for details about special rules.

give better PPL results. However, in Table 2, we can see that this is not always the case. For example, the *parent* and *opposite+parent* schemes are worse than *baseline* in the first iteration when λ=0.0, the *h-2+parent* and *h-2+opposite+parent* schemes are also worse than *h-2* scheme in the first iteration when λ=0.0.

Why wouldn't more information help? There are two possible reasons that come to mind:

1. Since the size of our training data is small (1M words), the data sparseness problem (over-parameterization) is more serious for the more complicated dependency structure. We can see the problem from Table 1: the NT/POS vocabularies grow much bigger as we enrich the NT/POS tags.

2. As mentioned in Section 3, a potential problem of enriching NT/POS tags in *parent* scheme is that pruning may delete some hypotheses at an early time and the search may not recover from those early mistakes. The result of this is a high parsing error and thus a worse language model.

Model	Iter=0	Iter=2
baseline	24.84	21.89
PA	29.00	22.63
OP	19.41	17.71
OP+PA	23.49	19.37
h-2	22.03	20.57
h-2+PA	19.64	18.20
h-2+OP	17.02	16.12
h-2+OP+PA	15.98	15.01

Table 5: PPL for training data

In order to validate the first hypothesis, we evaluated the training data PPL for each model scheme. As can be seen from Table 5, over-parameterization is indeed a problem. From scheme *h-2* to *h-2+opposite+parent*, as we add more information to the conditioning context, the training data PPL decreases. The test data PPL in Table 2 does not follow this trend, which is a clear sign of over-parameterization.

Over-parameterization might also occur for *parent* and *opposite+parent*, but it alone can not explain the high PPL of training data for both schemes. The LP/LR results in Table 3 show that bad parsing accuracy also plays a role in these situations. The labeled recall results of *parent* and *opposite+parent* are much worse than those of *baseline* and other schemes. The end-of-sentence parse completion strategy employed by the SLM is responsible for the high precision/low recall operation of the *parent* and *opposite+parent* models. Adding *h-2* remedies the parsing performance of the SLM in this situation, but not sufficiently.

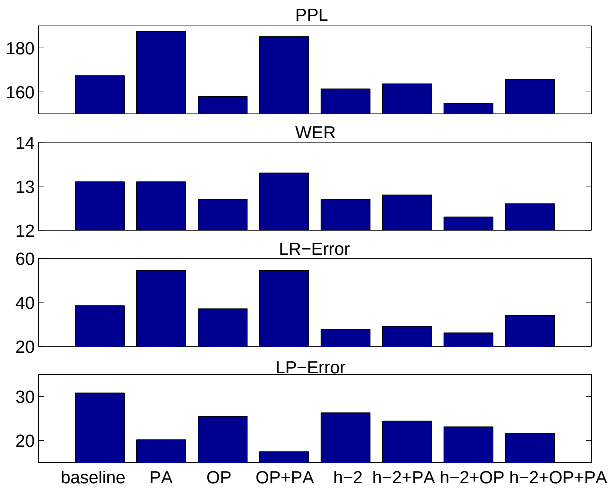


Figure 5: Comparison of PPL, WER(%), Labeled precision/recall(%) error

It is very interesting to note that labeled recall and language model performance (WER/PPL) are well correlated. Figure 5 compares PPL, WER (λ=0.0 at training iteration 0) and labeled precision/recall error(100-LP/LR) for all models. Overall, the labeled recall is well correlated with the WER and PPL values. Our results show that improvement in the parser accuracy is expected to lead to improvement in WER.

Finally, in comparison with the language model in (Roark, 2001) which is based on a probabilistic top-down parser, and with the Bihead/Trihead language models in (Charniak, 2001) which are based on immediate head parsing, our enriched models are less effective in reducing the test data PPL: the best PPL result of (Roark, 2001) on the same experimental setup is 137.3, and the best PPL result of (Charniak, 2001) is 126.1. We believe that examining the differences between the SLM and these models could help in understanding the degradation:

1. The parser in (Roark, 2001) uses a “pick-and-choose” strategy for the conditioning information used in the probability models. This allows the parser to choose information depending on the constituent that is being expanded. The SLM, on the other hand, always uses the same dependency structure that is decided beforehand.

2. The parser in (Charniak, 2001) is not a strict left-to-right parser. Since it is top-down, it is able to use the immediate head of a constituent before it occurs, while this immediate head is not available for conditioning by a strict left-to-right parser such as the SLM. Consequently, the interpolation with the 3-gram model is done at the sentence level, which is weaker than interpolating at the word level.

Since the WER results in (Roark, 2001) are based on less training data (2.2M words total), we do not have a fair comparison between our best model and Roark’s model.

5 Conclusion and Future Work

We have presented a study on enriching the syntactic dependency structures in the SLM. We have built and evaluated the performance of seven different models. All of our models improve on the baseline SLM in either PPL or WER or both. We have shown that adding the NT tag of the third most-recent exposed head in the parser model improves the parsing performance significantly. The improvement in parsing accuracy carries over to enhancing language model performance, as evaluated by both WER and PPL. Furthermore, our best result shows that an uninterpolated grammar-based language model can outperform a 3-gram model. The best model achieved an overall WER improvement of 10% relative to the 3-gram baseline.

Although conditioning on more contextual information helps, we should note that some of our models suffer from over-parameterization. One solution would be to apply the maximum entropy estimation technique (MaxEnt (Berger et al., 1996)) to all of the three components of the SLM, or at least to the CONSTRUCTOR. That would also allow for fine-tuning of the particular syntactic dependencies used in the model rather than the template based method we have used. Along these lines, the MaxEnt model has already shown promising improvements by combining syntactic dependencies in the WORD-PREDICTOR of the SLM (Wu and Khudanpur, 1999).

References

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

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of NAACL*, pages 132–139, Seattle, WA.

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting and 10th Conference of the European Chapter of ACL*, pages 116–123, Toulouse, France, July.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283–332, October.

Ciprian Chelba and Peng Xu. 2001. Richer syntactic dependencies for structured language modeling. In *Proceedings of the Automatic Speech Recognition and Understanding Workshop*, Madonna di Campiglio, Trento-Italy, December.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

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

Mark Johnson. 1998. Pcfg models of linguistic tree presentations. *Computational Linguistics*, 24(4):617–636.

Adwait Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *Second Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Providence, RI.

Brian Roark. 2001. *Robust Probabilistic Predictive Syntactic Processing: Motivations, Models and Applications*. Ph.D. thesis, Brown University, Providence, RI.

Jun Wu and Sanjeev Khudanpur. 1999. Combining nonlocal, syntactic and n-gram dependencies in language modeling. In *Proceedings of Eurospeech’99*, pages 2179–2182.