Efficient Structured Language Modeling for Speech Recognition

Ariya Rastrow, Mark Dredze, Sanjeev Khudanpur

Center for Language and Speech Processing, and
Human Language Technology Center of Excellence, Johns Hopkins University
`{ariya, mdredze, khudanpur}@jhu.edu`

Abstract

The structured language model (SLM) of [1] was one of the first to successfully integrate syntactic structure into language models. We extend the SLM framework in two new directions. First, we propose a new syntactic hierarchical interpolation that improves over previous approaches. Second, we develop a general information-theoretic algorithm for pruning the underlying Jelinek-Mercer interpolated LM used in [1], which substantially reduces the size of the LM, enabling us to train on large data. When combined with hill-climbing [2] the SLM is an accurate model, space-efficient and fast for rescoring large speech lattices. Experimental results on broadcast news demonstrate that the SLM outperforms a large 4-gram LM.

1. Introduction

Automatic Speech Recognition (ASR) relies on a language model (LM) to provide a strong linguistic prior on word sequences. Most language models rely on simple n-gram statistics and a wide range of smoothing and backoff techniques. Despite being simple and efficient, it is widely believed that limiting the context to only the $(n-1)$ most recent words ignores the structure of language, and several statistical frameworks have been proposed to incorporate the "syntactic structure of language back into language modeling" [1, 3, 4]. The Structured Language Model (SLM) [1] was one of the first successful attempts to build a statistical language model based on syntactic information. The SLM assigns a joint probability $P(W, T)$ to every word sequence W and every possible binary parse tree T, where T's terminals are words W with part-of-speech (POS) tags, and its internal nodes comprise non-terminal labels and lexical "heads" of phrases. Other approaches include using the exposed headwords in a maximum-entropy based LM [5], using exposed headwords from full-sentence parse tree in a neural network based LM [3], and the use of syntactic features in discriminative training [6]. However, most previous SLMs are large, complex and impractical for real-life applications.

We propose a new SLM based on dependency structures derived using a state-of-the-art probabilistic shift-reduce parser [7], in contrast to the original SLM which parametrizes the parser component with a conditional model. Two key improvements are also made to the SLM. First, we propose a simple hierarchical interpolation of syntactic parameters that achieves better performance without significant model complexity. Second, we maintain model efficiency on a large training corpus by using a general information-theoretic pruning method to reduce the size of the underlying Jelinek-Mercer LM [8]. When combined with fast hill-climbing rescoring [6], our framework is accurate, space efficient and practical for ASR.

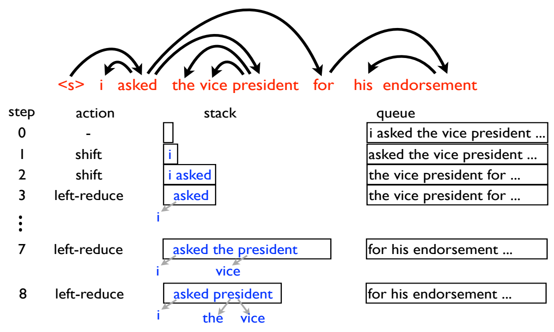

Figure 1: Actions of a shift-reduce parser to produce the dependency structure (up to the word `president`) shown above.

2. Dependency Language Models

Syntactic information can be encoded in terms of headwords and headtags of phrases, which are extracted from a syntactic analysis of a sentence [1, 3], such as a *dependency structure*. A dependency in a sentence holds between a *dependent* (or *modifier*) word and a *head* (or *governor*) word: the dependent *depends* on the head. These relations are encoded in a *dependency tree* (Figure 1), a directed graph where each edge (arc) encodes a head-dependent relation.

We use the shift-reduce incremental dependency parser of [7], which constructs a tree from sequence of transitions governed by a maximum-entropy classifier. Shift-reduce parsing places input words into a queue Q and partially built structures are organized by a stack S. Shift and reduce actions consume the queue and build the output parse on the stack. The classifier g assigns probabilities to each action, and the probability of a state $p_g(\pi)$ can be computed as the product of the probabilities of a sequence of actions that resulted in the state.

An incremental parser can provide partial syntactic analyses of the history at each word position. Parser states corresponding to the ith word w_i—states which have w_i as the top word in their queue— are *history-states*, denoted $\Pi_{-i} = \{\pi^0_{-i}, \pi^1_{-i} \cdots, \pi^{K_i}_{-i}\}$, where K_i is the total number of such states (others may be eliminated via beam pruning). Given Π_{-i} for all i, the probability assignment for w_i is:

$$p(w_i|W_{-i}) = \sum_{j=1}^{|\Pi_{-i}|} p\left(w_i|f(\pi^j_{-i}, W_{-i})\right) p_g(\pi^j_{-i}|W_{-i}) \quad (1)$$

where, W_{-i} is the word history $w_1, \ldots, w_{i-1}$ for w_i, π^j_{-i} is the j^{th} history-state of position i, $p_g(\pi^j_{-i}|W_{-i})$ is the probability assigned to state π^j_{-i} by the parser, and $f(\pi^j_{-i}, W_{-i})$ denotes an equivalence classification of the word-history and

$$\begin{tikzcd}
 & \texttt{HW :} & \texttt{HW+HT :} \\
f_M(\pi_{-i}, W_{-i}) \arrow[d] & (\text{h.w}_0, \text{h.w}_{-1}, \text{h.w}_{-2}) \arrow[d] & (\text{h.w}_0\text{h.t}_0, \text{h.w}_{-1}\text{h.t}_{-1}, \text{h.w}_{-2}\text{h.t}_{-2}) \arrow[d] \\
f_{M-1}(\pi_{-i}, W_{-i}) \arrow[d] & (\text{h.w}_0, \text{h.w}_{-1}) \arrow[d] & (\text{h.w}_0\text{h.t}_0, \text{h.w}_{-1}\text{h.t}_{-1}) \arrow[d] \\
f_{M-2}(\pi_{-i}, W_{-i}) \arrow[d] & (\text{h.w}_0) \arrow[d] & (\text{h.w}_0\text{h.t}_0) \arrow[d] \\
\vdots & () & ()
\end{tikzcd}$$

Figure 2: Examples of hierarchal interpolation schemes.

Language Model		Eval	Dev	
Kneser-Ney 4-gram		158	165	
SLM	`HW`	174	183	
	`HW+HT`	163	174	
	`HW+HT`$_2$	159	168	SLM Weight
Interp.	`HW`	149	154	0.35
	`HW+HT`	144	150	0.43
	`HW+HT`$_2$	**142**	**147**	0.48

Table 1: The preplexity of different `BN` language models.

parser history-state, capturing features from π^j_{-i} and W_{-i} that are most useful for predicting w_i. We will restrict $f(\pi^j_{-i}, W_{-i})$ to be based on only the heads of the partial trees $\{s_0\, s_1 \cdots\}$ in the stack. For example, in Figure 1, such a function applied to the parser state after step 8 yields the conditional probability $p(\texttt{for}|\texttt{president}, \texttt{NN})$ for predicting the word `for` (assuming the top headword `president` has POS tag `NN`). Given the choice of $f(.)$, the parameters of the model $p(w_i|f(\pi^j_{-i}, W_{-i}))$ are estimated to maximize the log-likelihood of the training data $\mathcal{T}$ using the Baum-Welch algorithm [9].

We use a hiearachy of fine-to-coarse equivalence classifications $f_M, f_{M-1}, \ldots, f_1$ to smooth probability estimates and handle unseen events via Jelinek-Mercer smoothing [8]. Let

$$f_M(\pi_{-i}, W_{-i}) \quad\rightarrow\quad f_{M-1}(\pi_{-i}, W_{-i}) \quad\rightarrow\quad \ldots \\ \rightarrow\quad f_2(\pi_{-i}, W_{-i}) \quad\rightarrow\quad f_1(\pi_{-i}, W_{-i}) \qquad (2)$$

be a set of M different equivalence classification of history state π_{-i}. Jelinek-Mercer smoothing utilizes linear interpolation of the ML estimated higher-order equivalence classification probabilities $p_{\text{ML}}\ (w|f_m)$ with lower order $p_{\text{ML}}\ (w|f_{m-1})$:

$$\begin{aligned} & p_{\text{interp}}(w_i|f_m(\pi_{-i}W_{-i})) \\ & \quad = \lambda_{f_m} p_{\text{ML}}(w_i|f_m(\pi_{-i}W_{-i})) \\ & \qquad +(1-\lambda_{f_m}) p_{\text{interp}}(w_i|f_{m-1}(\pi_{-i}W_{-i})), \end{aligned}$$

for $1 \leq m \leq M$, where the 0-th order model f_0 is a uniform distribution. Coefficients $\lambda_{f_m(\pi_{-i}W_{-i})}$ are estimated on a held-out set using the bucketing algorithm suggested in [10], which ties $\lambda_{f_m(\pi_{-i}W_{-i})}$'s based on the counts of $f_m(\pi_{-i}W_{-i})$'s on the training data. We use the expected count in $\mathcal{T}$ of $\mathbf{f_m}$ — a specific context feature in the set of all $f_m(\pi_{-i}W_{-i})$'s.

We perform the bucketing algorithm for each level $f_1, f_2, \cdots, f_M$ of equivalence classification separately, and estimate the bucketed $\lambda_{c(f_m)}$ using the Baum-Welch algorithm [9] to maximize the likelihood of held out data, where the word probability assignment in Eq. 1 is replaced with:

$$p(w_i|W_{-i}) = \sum_{j=1}^{|\Pi_i|} p_{\text{interp}}\left(w_i|f_M(\pi^j_{-i}, W_{-i})\right) p_g(\pi^j_{-i}|W_{-i})$$

We use two hierarchical interpolation schemes (Figure 2) for smoothing from [1], although we include larger contexts: equivalence classification of history-states by looking only at the headwords (`HW`) and head-tags (`HW+HT`) of the first 3 partial trees in the stack. "()" refers to an equivalence classification where no information is used from history (uni-grams.)

2.1. Preliminary Preplexity Results

Before proceeding to our new methods, we establish a baselined perplexity of the SLM using the above two hierarchical schemes. We consider *Broadcast News* (`BN`) recognition with the EARS BN03 corpus, which has $42M$ words of training text. rt04 ($45K$ words) is used for evaluation data. To interpolate our SLM with the baseline 4-gram model, we use rt03+dev04f (about $40K$ words) as development data. The vocabulary is $84K$ words. We sample about $20K$ sentences from the training text (excluded from training) to serve as heldout data for applying the bucketing algorithm and estimating λ's. For the dependency parser, data sets are converted to Treebank-style tokenization and POS-tagged using the tagger of [11]. Both the POS tagger and dependency parser are trained on the BN treebank from Ontonotes [12] and the WSJ Penn Treebank (converted to dependency trees), which consists of $1.2M$ tokens. Finally, we train a modified Kneser-Ney 4-gram LM on the tokenized training text to serve as the baseline LM.

Table 1 shows preplexities obtained for `BN` . Using headtags in the equivalence classifications used in the `HW+HT` interpolation scheme significantly improves the preplexity of the SLM. It also improves the performance of the interpolated LM.

3. Improved Hierarchical Interpolation

The original SLM hierarchical interpolation scheme is aggressive in that it drops both the tag and headword from the history. However, in many cases the headword's tag alone is sufficient, suggesting a more gradual interpolation. We propose a new interpolation where instead of dropping both the headword and headtag of a partial tree in the stack of a history-state at each level of the `HW+HT` hierarchical interpolation scheme, we first drop only the headword. E.g., in the `HW+HT` hierarchy of Figure 2, we use $(\text{h.w}_0\text{h.t}_0, \text{h.w}_{-1}\text{h.t}_{-1}, \text{h.w}_{-2}\text{h.t}_{-2}) \rightarrow (\text{h.w}_0\text{h.t}_0, \text{h.w}_{-1}\text{h.t}_{-1}, \text{h.t}_{-2}) \rightarrow (\text{h.w}_0\text{h.t}_0, \text{h.w}_{-1}\text{h.t}_{-1})$.

Keeping the headtag adds more specific information and at the same time is less sparse. We refer to this interpolation hierarchy as `HW+HT`$_2$. A similar idea is found, e.g., in the back-off hierarchical class n-gram language model [13]. Preplexity results (Table 1) confirm that the new scheme improves both interpolated and non-interpolated models.

4. Relative Entropy Pruning of the SLM

A significant drawback to the SLM is its size; the space of history-states Π_{-i} can be exponentially large. To reduce its size we prune the trained model $p_{\text{interp}}\ (w_i|f_M(\pi^j_{-i}, W_{-i}))$, taking into account accumulated training statistics, as opposed to pruning parser states locally. Our algorithm is based on the pruning scheme proposed in [14], which was used for pruning n-gram backoff language models. The goal is to minimize the *distance* between the implied distributions of the original and pruned models.

The m^{th}-level model in a hierarchical Jelinek-Mercer interpolated LM consists of the following parts:

1. $\mathbf{f_m} \in \{f_m\}$: The m^{th}-level contextual features which are

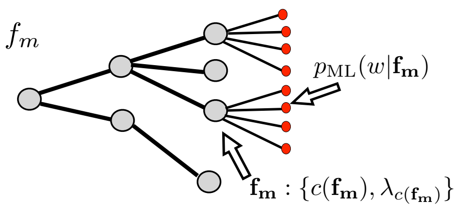
f_m
$p_{\text{ML}}(w|\mathbf{f_m})$
$\mathbf{f_m} : \{c(\mathbf{f_m}), \lambda_{c(\mathbf{f_m})}\}$

Figure 3: The data structure used for representing m^{th} level parameters of a Jelinek-Mercer LM.

observed in training.

2. $c(\mathbf{f_m})$: The (empirical expected) count of $\mathbf{f_m}$ in training.
3. $\lambda_{c(\mathbf{f_m})}$: The interpolation weights which are functions of context-feature counts and are used to interpolate m^{th}-level ML probabilities with the $(m-1)^{th}$-level probabilities.
4. $p_{\text{ML}}(w|\mathbf{f_m})$: Explicit maximum-likelihood probability of a word w given the context-feature $\mathbf{f_m}$.

The probability of w given the highest order context-feature $\mathbf{f_M}$ is calculated through the recursive interpolation scheme of Jelinek-Mercer smoothing:

$$\begin{aligned} & p_{\text{interp}}(w|\mathbf{f_m}) \\ & \quad = \lambda_{c(\mathbf{f_m})} p_{\text{ML}}(w|\mathbf{f_m}) + (1-\lambda_{c(\mathbf{f_m})}) p_{\text{interp}}(w|\mathbf{f_{m-1}}), \end{aligned}$$

for $1 \le m \le M$, where $\mathbf{f_M} \to \mathbf{f_{M-1}} \to \cdots \to \mathbf{f_1}$ represents the context-features used in each level of the hierarchical interpolation scheme. Figure 3 depicts the data structure used for representing m^{th} level parameters of a Jelinek-Mercer LM. The context-features, $\mathbf{f_m}$, and ML probabilities, $p_{\text{ML}}(w|\mathbf{f_m})$, are efficiently represented using a *trie*. We refer to pre-leaf and leaf nodes in the trie as "context-nodes" and "probability-nodes", respectively. For a given m^{th} level context-feature $\mathbf{f_m}$, let

$$\mathcal{S}(\mathbf{f_m}) = \{w | w \in V \text{ s.t } p_{\text{ML}}(w|\mathbf{f_m}) > 0\} \quad (3)$$

represent all the words in the vocabulary for which there exist an explicit ML estimated probability given the context-feature, i.e. $p_{\text{ML}}(w|\mathbf{f_m}) > 0$. In our Jelinek-Mercer pruning scheme, we want to remove those $\mathbf{f_m}$s and their corresponding explicit ML probabilities $p_{\text{ML}}(w|\mathbf{f_m})$ (for $w \in \mathcal{S}(\mathbf{f_m})$) which have the least impact on the LM distributions. Following [14] we use *Kullback-Leibler* (KL) divergence to measure the distance between the original and pruned LM. Let $p_{\text{interp}}(.|.)$ denote the original conditional probabilities and $p'_{\text{interp}}(.|.)$ the pruned ones. The KL divergence using m^{th} level probabilities is:

$$D(p||p') = \sum_{f_m,w} \hat{p}(f_m)\, p_{\text{interp}}(w|f_m) \log \frac{p_{\text{interp}}(w|f_m)}{p'_{\text{interp}}(w|f_m)}, \quad (4)$$

where $\hat{p}(f_m) = \frac{c(f_m)}{\sum_{f'_m} c(f_m)}$ is the empirical probably of a context-feature f_m. Ideally, we should minimize Eq. 4 over all possible subsets of $\mathbf{f_m}$ that are candidates for removal, but this is computationally intractable. Instead, we assume that each $\mathbf{f_m}$ effects $D(\cdot)$ independently and prune those with smallest impact. Removing $\mathbf{f_m}$ changes the model(s) as follows:

(i) For $w \in \mathcal{S}(\mathbf{f_m})$, the m^{th}-level probability is completely replaced by the lower-level interpolated model. That is,

$$\begin{aligned} p_{\text{interp}}(w|\mathbf{f_m}) &= \lambda_{c(\mathbf{f_m})} p_{\text{ML}}(w|\mathbf{f_M}) \\ &\quad + (1-\lambda_{c(\mathbf{f_m})}) p_{\text{interp}}(w|\mathbf{f_{m-1}}) \\ p'_{\text{interp}}(w|\mathbf{f_m}) &= p_{\text{interp}}(w|\mathbf{f_{m-1}}) \end{aligned} \quad (5)$$

Algorithm 1 Relative-entropy pruning of Jelinek-Mercer LM

```
θ: pruning threshold
$M_{\min}$: minimum hierarchical level for pruning
$T$: number of iterations
Algorithm:
for $iter$ in $1..T$ do
    for $m$ in $M..M_{\min}$ do
        for $\mathbf{f_m} \in f_m$ do
            calculate $D(p||p')$ due to removing $\mathbf{f_m}$ (Eq. 7)
            if $e^{D(p||p')} - 1 < \frac{\theta}{T}$
                prune $\mathbf{f_m}$ and make $c(\mathbf{f_m}) = 0$
        end for
    end for
    if anything pruned
        recompute buckets and λ's on the heldout data.
    else exit
end for
```

This is because when the context-feature $\mathbf{f_m}$ is removed, it is as if it has not been observed in the training data and $c(\mathbf{f_m})$ becomes zero. Therefore, $\lambda_{c(\mathbf{f_m})} = 0$.

(ii) For $w \notin \mathcal{S}(\mathbf{f_m})$, $p_{\text{ML}}(w|\mathbf{f_M})$ is zero, so that

$$\begin{aligned} p_{\text{interp}}(w|\mathbf{f_m}) &= (1-\lambda_{c(\mathbf{f_m})}) p_{\text{interp}}(w|\mathbf{f_{m-1}}) \\ p'_{\text{interp}}(w|\mathbf{f_m}) &= p_{\text{interp}}(w|\mathbf{f_{m-1}}). \end{aligned} \quad (6)$$

(iii) Since we assume $c(\mathbf{f_m}) = 0$ after removing the context-node $\mathbf{f_m}$, the buckets used to tie λ's—and therefore, the maximum-likelihood estimates of λ's on the heldout data—will be affected. This could affect $p'_{\text{interp}}(w|\mathbf{f'_m})$ even for $\mathbf{f'_m} \neq \mathbf{f_m}$ and is hence computationally difficult to measure. Therefore, we use an iterative algorithm: each iteration measures only the changes in $D(p||p')$ due to effects (i) and (ii), ignoring (iii). After pruning $\mathbf{f_m}$'s, we run the bucketing algorithm and re-estimate the λs for the new pruned LM on held out data.

By ignoring (iii) all estimates not involving context-feature $\mathbf{f_m}$ remain unchanged, so the KL divergence from removing $\mathbf{f_m}$ is:

$$\begin{aligned} D(p||p') = p(\mathbf{f_m}) \Bigg(& \sum_{w \in \mathcal{S}(\mathbf{f_m})} p_{\text{interp}}(w|\mathbf{f_m}) \log \frac{p_{\text{interp}}(w|\mathbf{f_m})}{p'_{\text{interp}}(w|\mathbf{f_m})} \\ & + \sum_{w \notin \mathcal{S}(\mathbf{f_m})} p_{\text{interp}}(w|\mathbf{f_m}) \log \frac{p_{\text{interp}}(w|\mathbf{f_m})}{p'_{\text{interp}}(w|\mathbf{f_m})} \Bigg) \end{aligned}$$

For the first and second sum we use probabilities in Eq. 5 and Eq. 6, respectively. The above thus simplifies to

$$\begin{aligned} D(p||p') = p(\mathbf{f_m}) \Bigg(& \sum_{w \in \mathcal{S}(\mathbf{f_m})} p_{\text{interp}}(w|\mathbf{f_m}) \log(A) \\ & + \left[1 - \sum_{w \in \mathcal{S}(\mathbf{f_m})} p_{\text{interp}}(w|\mathbf{f_m}) \right] \log(1-\lambda_{c(\mathbf{f_m})}) \Bigg) \end{aligned} \quad (7)$$

where the term A is,

$$A = \lambda_{c(\mathbf{f_m})} \frac{p_{\text{ML}}(w|\mathbf{f_m})}{p_{\text{interp}}(w|\mathbf{f_{m-1}})} + (1-\lambda_{c(\mathbf{f_m})})$$

Note that a single iteration over all $w \in \mathcal{S}(\mathbf{f_m})$ is needed to calculate both the first sum in Eq. 7 and $\sum_{w \in \mathcal{S}(\mathbf{f_m})} p_{\text{interp}}(w|\mathbf{f_m})$.

Algorithm 1 summarizes our proposed pruning procedure. In each iteration, we iterate over all context-features, from level M to $M_{\min}$, and remove nodes for which $(e^{D(p||p')} - 1) < \frac{\theta}{T}$. Dividing the threshold θ by number of iterations reduces pruning in each iteration and distributes pruning across T iterations, where we re-estimate λ's after each iteration to take into account changes due to (iii) above.

Threshold	Eval	Dev	# Param.	Mem. Size
SLM				
0	159	168	413.5 M	12.5 GB
1×10^{-7}	159	175	143.3 M	4.4 GB
5×10^{-7}	161	183	90 M	2.8 GB
Interpolated				
0	142	147	(413.5+39) M	(12.5+1.7) GB
1×10^{-7}	141	148	(143.3+39) M	(4.4+1.7) GB
5×10^{-7}	141	149	(90+39) M	(2.8+1.7) GB

Table 2: Preplexity and memory size of the $\mathtt{HW{+}HT_2}$ SLMs.

4.1. Pruning Experiments and Results

We evaluate the effect of pruning using the setup described in Section 2.1. We apply pruning to our best model, the Jelinek-Mercer SLM with $\mathtt{HW{+}HT_2}$, with $M = 7, T = 5$ and $M_{\min} = 4$. The lowest hierarchy level that we considered for pruning is $(\mathrm{h.w_0 h.t_0}, \mathrm{h.t_{-1}})$. We prune the non-interpolated SLM and then interpolate it with the Kneser-Ney 4-gram model. Our goal is to reduce the SLM to a comparable size to the baseline model, which stores $39M$ n-grams in 1.7GB memory.

Pruning reduces the size of the SLMs significantly, as seen in Table 2. SLM size drops from 12.5GB to 2.8GB without much loss in test-set perplexity. In fact, when interpolated with the Kneser-Ney 4-gram, the Dev-data perplexity is almost unchanged. For the Eval-data, the performance ever so slightly better, possibly due to over-fitting of the unpruned LM.

5. Speech Recognition Experiments

We conclude with experiments using our improved SLM for speech recognition. Since the SLM uses equivalence classification of histories based on dependency structures, it is a long-span model, which excludes the ability for direct lattice rescoring. Therefore we use the SLM as the long-span model in the hill-climbing algorithm of [2], which is demonstrably better than N-best rescoring.

Since the hill climbing algorithm is a greedy search with no guarantees of global optimality, we use 20 random-restarts, repeating hill climbing with different initial word sequences sampled from the speech lattice, and output the best scoring final output. The hypotheses in the original lattices are not in Treebank-style tokenization, so we tokenize each hypothesis during hill climbing to the Treebank style for POS-tagging, dependency parsing and SLM scoring. The final output is converted back to standard tokenization for WER computation.

We use the SLM with $\mathtt{HW{+}HT_2}$ interploated with the baseline 4-gram LM. Lattices are generated using a pruned Kneserney 4-gram LM (trained on untokenized training text.) The lattice generating LM has 15.4% WER and our baseline 4-gram LM (unpruned) has 15.1% on the evaluation data. We consider both the unpruned and pruned ($\theta = 5 \times 10^{-7}$) SLM.

The recognition results in Table 3 show that the interpolated SLM significantly improves the baseline performance: a 0.6% absolute improvement in WER. In addition, our proposed pruning method does not degrade performance, despite significantly reducing LM size. Finally, we observe that hill-climbing with 1 and 20 initial paths results in an average of only 40 and 300 hypotheses evaluated per utterance, a big saving over N-best rescoring schemes traditionally employed for long-span language models.

LM		Pruned	Unpruned
Kneser-Ney 4-gram		15.4%	15.1%
Interp. SLM	1 initial path	14.7%	14.7%
	20 initial paths	**14.5%**	**14.5%**

Table 3: Word error rates for speech recognition using the proposed SLM with $\mathtt{HW{+}HT_2}$ in the hill-climbing algorithm.

6. Conclusions

This paper presents useful extensions to the original structured language model to make it more accurate, space efficient and practical for speech recognition. It introduces a simple hierarchical interpolation scheme which improves the preplexity of the SLM. Furthermore, a general information-theoretic pruning algorithm is developed to reduce model size in the Jelinek-Mercer LM smoothing framework. Experiments indicate that this extended SLM framework, when combined with an efficient hill-climbing algorithm, improves recognition accuracy over regular n-gram models.

7. References

[1] C. Chelba and F. Jelinek, “Structured language modeling,” *Computer Speech and Language*, vol. 14, no. 4, pp. 283–332, 2000.

[2] A. Rastrow, M. Dreyer, A. Sethy, S. Khudanpur, B. Ramabhadran, and M. Dredze, “Hill climbing on speech lattices : A new rescoring framework,” in *Proceeding of ICASSP*, 2011.

[3] H. K. J. Kuo, L. Mangu, A. Emami, I. Zitouni, and L. Young-Suk, “Syntactic features for Arabic speech recognition,” in *Proc. ASRU*, 2009.

[4] M. Collins, B. Roark, and M. Saraclar, “Discriminative syntactic language modeling for speech recognition,” in *ACL*, 2005.

[5] S. Khudanpur and J. Wu, “Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling,” *Computer Speech and Language*, pp. 355–372, 2000.

[6] A. Rastrow, M. Dredze, and S. Khudanpur, “Efficient discrimnative training of long-span language models,” in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011.

[7] K. Sagae and J. Tsujii, “Dependency parsing and domain adaptation with LR models and parser ensembles,” in *Proc. EMNLP-CoNLL*, vol. 7, 2007, pp. 1044–1050.

[8] F. Jelinek and R. L. Mercer, “Interpolated estimation of Markov source parameters from sparse data,” in *Proceedings of the Workshop on Pattern Recognition in Practice*, 1980, pp. 381–397.

[9] L. E. Baum, “An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes,” *Inequalities*, vol. 3, pp. 1–8, 1972.

[10] L. Bahl, “A maximum likelihood approach to continuous speech recognition,” *IEEE Transactions on Pattern Analysis and Machine Inteligence (PAMI)*, vol. 5, no. 2, pp. 179–190, 1983.

[11] Y. Tsuruoka, Y. Miyao, and J. Kazama, “Learning with Lookahead : Can History-Based Models Rival Globally Optimized Models ?” in *Proc. CoNLL*, no. June, 2011, pp. 238–246.

[12] R. Weischedel, S. Pradhan, L. Ramshaw, M. Palmer, N. Xue, M. Marcus, A. Taylor, C. Greenberg, E. Hovy, R. Belvin, and A. Houston, *OntoNotes Release 2.0*, Linguistic Data Consortium, Philadelphia, 2008.

[13] I. Zitouni, “Backoff hierarchical class n-gram language models: effectiveness to model unseen events in speech recognition,” *Computer Speech & Language*, vol. 21, no. 1, pp. 88–104, 2007.

[14] A. Stolcke, “Entropy-based pruning of backoff language models,” in *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, 1998.