Discriminative Syntactic Language Modeling for Speech Recognition

Michael Collins
MIT CSAIL
mcollins@csail.mit.edu

Brian Roark
OGI/OHSU
roark@cslu.ogi.edu

Murat Saraclar
Bogazici University
murat.saraclar@boun.edu.tr

Abstract

We describe a method for discriminative training of a language model that makes use of syntactic features. We follow a *reranking* approach, where a baseline recogniser is used to produce 1000-best output for each acoustic input, and a second "reranking" model is then used to choose an utterance from these 1000-best lists. The reranking model makes use of syntactic features together with a parameter estimation method that is based on the perceptron algorithm. We describe experiments on the Switchboard speech recognition task. The syntactic features provide an additional 0.3% reduction in test–set error rate beyond the model of (Roark et al., 2004a; Roark et al., 2004b) (significant at $p < 0.001$), which makes use of a discriminatively trained n-gram model, giving a total reduction of 1.2% over the baseline Switchboard system.

1 Introduction

The predominant approach within language modeling for speech recognition has been to use an n-gram language model, within the "source-channel" or "noisy-channel" paradigm. The language model assigns a probability $P_l(\mathbf{w})$ to each string $\mathbf{w}$ in the language; the acoustic model assigns a conditional probability $P_a(\mathbf{a}|\mathbf{w})$ to each pair $(\mathbf{a}, \mathbf{w})$ where $\mathbf{a}$ is a sequence of acoustic vectors, and $\mathbf{w}$ is a string. For a given acoustic input $\mathbf{a}$, the highest scoring string under the model is

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} (\beta \log P_l(\mathbf{w}) + \log P_a(\mathbf{a}|\mathbf{w})) \quad (1)$$

where $\beta > 0$ is some value that reflects the relative importance of the language model; β is typically chosen by optimization on held-out data. In an n-gram language model, a Markov assumption is made, namely that each word depends only on the previous $(n-1)$ words. The parameters of the language model are usually estimated from a large quantity of text data. See (Chen and Goodman, 1998) for an overview of estimation techniques for n-gram models.

This paper describes a method for incorporating syntactic features into the language model, using discriminative parameter estimation techniques. We build on the work in Roark et al. (2004a; 2004b), which was summarized and extended in Roark et al. (2005). These papers used discriminative methods for n-gram language models. Our approach reranks the 1000-best output from the Switchboard recognizer of Ljolje et al. (2003).[1] Each candidate string $\mathbf{w}$ is parsed using the statistical parser of Collins (1999) to give a parse tree $\mathcal{T}(\mathbf{w})$. Information from the parse tree is incorporated in the model using a feature-vector approach: we define $\Phi(\mathbf{a}, \mathbf{w})$ to be a d-dimensional feature vector which in principle could track arbitrary features of the string $\mathbf{w}$ together with the acoustic input $\mathbf{a}$. In this paper we restrict $\Phi(\mathbf{a}, \mathbf{w})$ to only consider the string $\mathbf{w}$ and/or the parse tree $\mathcal{T}(\mathbf{w})$ for $\mathbf{w}$. For example, $\Phi(\mathbf{a}, \mathbf{w})$ might track counts of context-free rule productions in $\mathcal{T}(\mathbf{w})$, or bigram lexical dependencies within $\mathcal{T}(\mathbf{w})$. The optimal string under our new model is defined as

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} (\beta \log P_l(\mathbf{w}) + \langle \bar{\alpha}, \Phi(\mathbf{a}, \mathbf{w}) \rangle + \log P_a(\mathbf{a}|\mathbf{w})) \quad (2)$$

where the $\arg\max$ is taken over all strings in the 1000-best list, and where $\bar{\alpha} \in \mathbb{R}^d$ is a parameter vector specifying the "weight" for each feature in Φ (note that we define $\langle x, y \rangle$ to be the inner, or dot

[1] Note that (Roark et al., 2004a; Roark et al., 2004b) give results for an n-gram approach on this data which makes use of both lattices and 1000-best lists. The results on 1000-best lists were very close to results on lattices for this domain, suggesting that the 1000-best approximation is a reasonable one.

Proceedings of the 43rd Annual Meeting of the ACL, pages 507–514,
Ann Arbor, June 2005.

product, between vectors x and y). For this paper, we train the parameter vector $\bar{\alpha}$ using the perceptron algorithm (Collins, 2004; Collins, 2002). The perceptron algorithm is a very fast training method, in practice requiring only a few passes over the training set, allowing for a detailed comparison of a wide variety of feature sets.

A number of researchers have described work that incorporates syntactic language models into a speech recognizer. These methods have almost exclusively worked within the noisy channel paradigm, where the syntactic language model has the task of modeling a distribution over strings in the language, in a very similar way to traditional n-gram language models. The Structured Language Model (Chelba and Jelinek, 1998; Chelba and Jelinek, 2000; Chelba, 2000; Xu et al., 2002; Xu et al., 2003) makes use of an incremental shift-reduce parser to enable the probability of words to be conditioned on k previous c-commanding lexical heads, rather than simply on the previous k words. Incremental top-down and left-corner parsing (Roark, 2001a; Roark, 2001b) and head-driven parsing (Charniak, 2001) approaches have directly used generative PCFG models as language models. In the work of Wen Wang and Mary Harper (Wang and Harper, 2002; Wang, 2003; Wang et al., 2004), a constraint dependency grammar and a finite-state tagging model derived from that grammar were used to exploit syntactic dependencies.

Our approach differs from previous work in a couple of important respects. First, through the feature-vector representations $\Phi(\mathbf{a}, \mathbf{w})$ we can essentially incorporate arbitrary sources of information from the string or parse tree into the model. We would argue that our method allows considerably more flexibility in terms of the choice of features in the model; in previous work features were incorporated in the model through modification of the underlying generative parsing or tagging model, and modifying a generative model is a rather indirect way of changing the features used by a model. In this respect, our approach is similar to that advocated in Rosenfeld et al. (2001), which used Maximum Entropy modeling to allow for the use of shallow syntactic features for language modeling.

A second contrast between our work and previous work, including that of Rosenfeld et al. (2001), is in the use of discriminative parameter estimation techniques. The criterion we use to optimize the parameter vector $\bar{\alpha}$ is closely related to the end goal in speech recognition, i.e., word error rate. Previous work (Roark et al., 2004a; Roark et al., 2004b) has shown that discriminative methods within an n-gram approach can lead to significant reductions in WER, in spite of the features being of the same type as the original language model. In this paper we extend this approach, by including syntactic features that were not in the baseline speech recognizer.

This paper describe experiments using a variety of syntactic features within this approach. We tested the model on the Switchboard (SWB) domain, using the recognizer of Ljolje et al. (2003). The discriminative approach for n-gram modeling gave a 0.9% reduction in WER on this domain; the syntactic features we describe give a further 0.3% reduction.

In the remainder of this paper, section 2 describes previous work, including the parameter estimation methods we use, and section 3 describes the feature-vector representations of parse trees that we used in our experiments. Section 4 describes experiments using the approach.

2 Background

2.1 Previous Work

Techniques for exploiting stochastic context-free grammars for language modeling have been explored for more than a decade. Early approaches included algorithms for efficiently calculating string prefix probabilities (Jelinek and Lafferty, 1991; Stolcke, 1995) and approaches to exploit such algorithms to produce n-gram models (Stolcke and Segal, 1994; Jurafsky et al., 1995). The work of Chelba and Jelinek (Chelba and Jelinek, 1998; Chelba and Jelinek, 2000; Chelba, 2000) involved the use of a shift-reduce parser trained on Penn treebank style annotations, that maintains a weighted set of parses as it traverses the string from left-to-right. Each word is predicted by each candidate parse in this set at the point when the word is shifted, and the conditional probability of the word given the previous words is taken as the weighted sum of the conditional probabilities provided by each parse. In this approach, the probability of a word is conditioned by the top two lexical heads on the stack of the par-

ticular parse. Enhancements in the feature set and improved parameter estimation techniques have extended this approach in recent years (Xu et al., 2002; Xu et al., 2003).

Roark (2001a; 2001b) pursued a different derivation strategy from Chelba and Jelinek, and used the parse probabilities directly to calculate the string probabilities. This work made use of a left-to-right, top-down, beam-search parser, which exploits rich lexico-syntactic features from the left context of each derivation to condition derivation move probabilities, leading to a very peaked distribution. Rather than normalizing a prediction of the next word over the beam of candidates, as in Chelba and Jelinek, in this approach the string probability is derived by simply summing the probabilities of all derivations for that string in the beam.

Other work on syntactic language modeling includes that of Charniak (2001), which made use of a non-incremental, head-driven statistical parser to produce string probabilities. In the work of Wen Wang and Mary Harper (Wang and Harper, 2002; Wang, 2003; Wang et al., 2004), a constraint dependency grammar and a finite-state tagging model derived from that grammar, were used to exploit syntactic dependencies. The processing advantages of the finite-state encoding of the model has allowed for the use of probabilities calculated off-line from this model to be used in the first pass of decoding, which has provided additional benefits. Finally, Och et al. (2004) use a reranking approach with syntactic information within a machine translation system.

Rosenfeld et al. (2001) investigated the use of syntactic features in a Maximum Entropy approach. In their paper, they used a shallow parser to annotate base constituents, and derived features from sequences of base constituents. The features were indicator features that were either (1) exact matches between a set or sequence of base constituents with those annotated on the hypothesis transcription; or (2) tri-tag features from the constituent sequence. The generative model that resulted from their feature set resulted in only a very small improvement in either perplexity or word-error-rate.

2.2 Global Linear Models

We follow the framework of Collins (2002; 2004), recently applied to language modeling in Roark et al. (2004a; 2004b). The model we propose consists of the following components:

- $\mathbf{GEN}(\mathbf{a})$ is a set of candidate strings for an acoustic input $\mathbf{a}$. In our case, $\mathbf{GEN}(\mathbf{a})$ is a set of 1000-best strings from a first-pass recognizer.
- $\mathcal{T}(\mathbf{w})$ is the parse tree for string $\mathbf{w}$.
- $\Phi(\mathbf{a}, \mathbf{w}) \in \mathbb{R}^d$ is a feature-vector representation of an acoustic input $\mathbf{a}$ together with a string $\mathbf{w}$.
- $\bar{\alpha} \in \mathbb{R}^d$ is a parameter vector.
- The output of the recognizer for an input $\mathbf{a}$ is defined as

$$F(\mathbf{a}) = \underset{\mathbf{w} \in \mathbf{GEN}(\mathbf{a})}{\operatorname{argmax}} \langle \Phi(\mathbf{a}, \mathbf{w}), \bar{\alpha} \rangle \quad (3)$$

In principle, the feature vector $\Phi(\mathbf{a}, \mathbf{w})$ could take into account any features of the acoustic input $\mathbf{a}$ together with the utterance $\mathbf{w}$. In this paper we make a couple of restrictions. First, we define the first feature to be

$$\Phi_1(\mathbf{a}, \mathbf{w}) = \beta \log P_l(\mathbf{w}) + \log P_a(\mathbf{a}|\mathbf{w})$$

where $P_l(\mathbf{w})$ and $P_a(\mathbf{a}|\mathbf{w})$ are language and acoustic model scores from the baseline speech recognizer. In our experiments we kept β fixed at the value used in the baseline recogniser. It can then be seen that our model is equivalent to the model in Eq. 2. Second, we restrict the remaining features $\Phi_2(\mathbf{a}, \mathbf{w}) \ldots \Phi_d(\mathbf{a}, \mathbf{w})$ to be sensitive to the string $\mathbf{w}$ alone.[2] In this sense, the scope of this paper is limited to the language modeling problem. As one example, the language modeling features might take into account n-grams, for example through definitions such as

$$\Phi_2(\mathbf{a}, \mathbf{w}) = \text{Count of } \textit{the the} \text{ in } \mathbf{w}$$

Previous work (Roark et al., 2004a; Roark et al., 2004b) considered features of this type. In this paper, we introduce syntactic features, which may be sensitive to the parse tree for $\mathbf{w}$, for example

$$\Phi_3(\mathbf{a}, \mathbf{w}) = \text{Count of } \mathsf{S} \rightarrow \mathsf{NP}\ \mathsf{VP} \text{ in } \mathcal{T}(\mathbf{w})$$

where $\mathsf{S} \rightarrow \mathsf{NP}\ \mathsf{VP}$ is a context-free rule production. Section 3 describes the full set of features used in the empirical results presented in this paper.

[2] Future work may consider features of the acoustic sequence $\mathbf{a}$ together with the string $\mathbf{w}$, allowing the approach to be applied to acoustic modeling.

2.2.1 Parameter Estimation

We now describe how the parameter vector $\bar{\alpha}$ is estimated from a set of training utterances. The training set consists of examples $(\mathbf{a}_i, \mathbf{w}_i)$ for $i = 1 \ldots m$, where $\mathbf{a}_i$ is the i'th acoustic input, and $\mathbf{w}_i$ is the transcription of this input. We briefly review the two training algorithms described in Roark et al. (2004b), the perceptron algorithm and global conditional log-linear models (GCLMs).

Figure 1 shows the perceptron algorithm. It is an online algorithm, which makes several passes over the training set, updating the parameter vector after each training example. For a full description of the algorithm, see Collins (2004; 2002).

A second parameter estimation method, which was used in (Roark et al., 2004b), is to optimize the log-likelihood under a log-linear model. Similar approaches have been described in Johnson et al. (1999) and Lafferty et al. (2001). The objective function used in optimizing the parameters is

$$L(\bar{\alpha}) = \sum_i \log P(\mathbf{s}_i|\mathbf{a}_i, \bar{\alpha}) - C \sum_j \alpha_j^2 \quad (4)$$

where $P(\mathbf{s}_i|\mathbf{a}_i, \bar{\alpha}) = \frac{e^{\langle \Phi(\mathbf{a}_i, \mathbf{s}_i), \bar{\alpha} \rangle}}{\sum_{\mathbf{w} \in \mathbf{GEN}(\mathbf{a}_i)} e^{\langle \Phi(\mathbf{a}_i, \mathbf{w}), \bar{\alpha} \rangle}}$.

Here, each $\mathbf{s}_i$ is the member of $\mathbf{GEN}(\mathbf{a}_i)$ which has lowest WER with respect to the target transcription $\mathbf{w}_i$. The first term in $L(\bar{\alpha})$ is the log-likelihood of the training data under a conditional log-linear model. The second term is a regularization term which penalizes large parameter values. C is a constant that dictates the relative weighting given to the two terms. The optimal parameters are defined as

$$\bar{\alpha}^* = \arg\max_{\bar{\alpha}} L(\bar{\alpha})$$

We refer to these models as global conditional log-linear models (GCLMs).

Each of these algorithms has advantages. A number of results—e.g., in Sha and Pereira (2003) and Roark et al. (2004b)—suggest that the GCLM approach leads to slightly higher accuracy than the perceptron training method. However the perceptron converges very quickly, often in just a few passes over the training set—in comparison GCLM's can take tens or hundreds of gradient calculations before convergence. In addition, the perceptron can be used as an effective feature selection technique, in that at each training example it only increments features seen on $\mathbf{s}_i$ or $\mathbf{y}_i$, effectively ignoring all other features seen on members of $\mathbf{GEN}(\mathbf{a}_i)$. For example, in the experiments in Roark et al. (2004a), the perceptron converged in around 3 passes over the training set, while picking non-zero values for around 1.4 million n-gram features out of a possible 41 million n-gram features seen in the training set.

Input: A parameter specifying the number of iterations over the training set, T. A value for the first parameter, α. A feature-vector representation $\Phi(\mathbf{a}, \mathbf{w}) \in \mathbb{R}^d$. Training examples $(\mathbf{a}_i, \mathbf{w}_i)$ for $i = 1 \ldots m$. An n-best list $\mathbf{GEN}(\mathbf{a}_i)$ for each training utterance. We take $\mathbf{s}_i$ to be the member of $\mathbf{GEN}(\mathbf{a}_i)$ which has the lowest WER when compared to $\mathbf{w}_i$.

Initialization: Set $\alpha_1 = \alpha$, and $\alpha_j = 0$ for $j = 2 \ldots d$.

Algorithm: For $t = 1 \ldots T, i = 1 \ldots m$

•Calculate $\mathbf{y}_i = \arg\max_{\mathbf{w} \in \mathbf{GEN}(\mathbf{a}_i)} \langle \Phi(\mathbf{a}_i, \mathbf{w}), \bar{\alpha} \rangle$

• For $j = 2 \ldots m$, set $\bar{\alpha}_j = \bar{\alpha}_j + \Phi_j(\mathbf{a}_i, \mathbf{s}_i) - \Phi_j(\mathbf{a}_i, \mathbf{y}_i)$

Output: Either the final parameters $\bar{\alpha}$, or the averaged parameters $\bar{\alpha}_{avg}$ defined as $\bar{\alpha}_{avg} = \sum_{t,i} \bar{\alpha}^{t,i}/mT$ where $\bar{\alpha}^{t,i}$ is the parameter vector after training on the i'th training example on the t'th pass through the training data.

Figure 1: The perceptron training algorithm. Following Roark et al. (2004a), the parameter α_1 is set to be some constant α that is typically chosen through optimization over the development set. Recall that α_1 dictates the weight given to the baseline recognizer score.

For the present paper, to get a sense of the relative effectiveness of various kinds of syntactic features that can be derived from the output of a parser, we are reporting results using just the perceptron algorithm. This has allowed us to explore more of the potential feature space than we would have been able to do using the more costly GCLM estimation techniques. In future we plan to apply GLCM parameter estimation methods to the task.

3 Parse Tree Features

We tagged each candidate transcription with (1) part-of-speech tags, using the tagger documented in Collins (2002); and (2) a full parse tree, using the parser documented in Collins (1999). The models for both of these were trained on the Switchboard

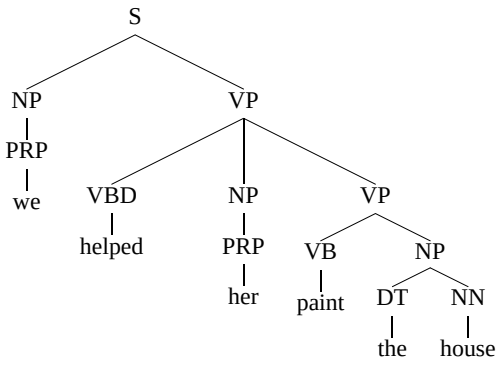

Figure 2: An example parse tree

(a)
we/PRP helped/VBD her/PRP paint/VB the/DT house/NN
(b)
we/NP^b helped/VP^b her/NP^b paint/VP^b the/NP^b house/NP^c
(c)
we/PRP-NP^b helped/VBD-VP^b her/PRP-NP^b paint/VB-VP^b the/DT-NP^b house/NN-NP^c
(d)
we/NP helped/VP her/NP paint/VP house/NP

Figure 3: Sequences derived from a parse tree: (a) POS-tag sequence; (b) Shallow parse tag sequence—the superscripts b and c refer to the beginning and continuation of a phrase respectively; (c) Shallow parse tag plus POS tag sequence; and (d) Shallow category with lexical head sequence

treebank, and applied to candidate transcriptions in both the training and test sets. Each transcription received one POS-tag annotation and one parse tree annotation, from which features were extracted.

Figure 2 shows a Penn Treebank style parse tree that is of the sort produced by the parser. Given such a structure, there is a tremendous amount of flexibility in selecting features. The first approach that we follow is to map each parse tree to sequences encoding part-of-speech (POS) decisions, and "shallow" parsing decisions. Similar representations have been used by (Rosenfeld et al., 2001; Wang and Harper, 2002). Figure 3 shows the sequential representations that we used. The first simply makes use of the POS tags for each word. The latter representations make use of sequences of non-terminals associated with lexical items. In 3(b), each word in the string is associated with the beginning or continuation of a shallow phrase or "chunk" in the tree. We include any non-terminals above the level of POS tags as potential chunks: a new "chunk" (VP, NP, PP etc.) begins whenever we see the initial word of the phrase dominated by the non-terminal. In 3(c), we show how POS tags can be added to these sequences. The final type of sequence mapping, shown in 3(d), makes a similar use of chunks, but preserves only the headword seen with each chunk.[3]

From these sequences of categories, various features can be extracted, to go along with the n-gram features used in the baseline. These include n-tag features, e.g. $t_{i-2}t_{i-1}t_i$ (where t_i represents the tag in position i); and composite tag/word features, e.g. t_iw_i (where w_i represents the word in position i) or, more complicated configurations, such as $t_{i-2}t_{i-1}w_{i-1}t_iw_i$. These features can be extracted from whatever sort of tag/word sequence we provide for feature extraction, e.g. POS-tag sequences or shallow parse tag sequences.

One variant that we performed in feature extraction had to do with how speech repairs (identified as EDITED constituents in the Switchboard style parse trees) and filled pauses or interjections (labeled with the INTJ label) were dealt with. In the simplest version, these are simply treated like other constituents in the parse tree. However, these can disrupt what may be termed the *intended* sequence of syntactic categories in the utterance, so we also tried skipping these constituents when mapping from the parse tree to shallow parse sequences.

The second set of features we employed made use of the full parse tree when extracting features. For this paper, we examined several features templates of this type. First, we considered context-free rule instances, extracted from each local node in the tree. Second, we considered features based on lexical heads within the tree. Let us first distinguish between POS-tags and non-POS non-terminal categories by calling these latter constituents NTs. For each constituent NT in the tree, there is an associated lexical head (H_{NT}) and the POS-tag of that lexical head (HP_{NT}). Two simple features are NT/H_{NT} and NT/HP_{NT} for every NT constituent in the tree.

[3]It should be noted that for a very small percentage of hypotheses, the parser failed to return a full parse tree. At the end of every shallow tag or category sequence, a special end of sequence tag/word pair "</parse> </parse>"was emitted. In contrast, when a parse failed, the sequence consisted of solely "<noparse> <noparse>".

Feature	Examples from figure 2
(P,HC_P,C_i,{+,-}{1,2},H_P,H_{C_i})	(VP,VB,NP,1,paint,house) (S,VP,NP,-1,helped,we)
(P,HC_P,C_i,{+,-}{1,2},H_P,HP_{C_i})	(VP,VB,NP,1,paint,NN) (S,VP,NP,-1,helped,PRP)
(P,HC_P,C_i,{+,-}{1,2},HP_P,H_{C_i})	(VP,VB,NP,1,VB,house) (S,VP,NP,-1,VBD,we)
(P,HC_P,C_i,{+,-}{1,2},HP_P,HP_{C_i})	(VP,VB,NP,1,VB,NN) (S,VP,NP,-1,VBD,PRP)

Table 1: Examples of head-to-head features. The examples are derived from the tree in figure 2.

Using the heads as identified in the parser, example features from the tree in figure 2 would be S/VBD, S/helped, NP/NN, and NP/house.

Beyond these constituent/head features, we can look at the head-to-head dependencies of the sort used by the parser. Consider each local tree, consisting of a parent node (P), a head child (HC_P), and k non-head children ($C_1 \ldots C_k$). For each non-head child C_i, it is either to the left or right of HC_P, and is either adjacent or non-adjacent to HC_P. We denote these positional features as an integer, positive if to the right, negative if to the left, 1 if adjacent, and 2 if non-adjacent. Table 1 shows four head-to-head features that can be extracted for each non-head child C_i. These features include dependencies between pairs of lexical items, between a single lexical item and the part-of-speech of another item, and between pairs of part-of-speech tags in the parse.

4 Experiments

The experimental set-up we use is very similar to that of Roark et al. (2004a; 2004b), and the extensions to that work in Roark et al. (2005). We make use of the Rich Transcription 2002 evaluation test set (rt02) as our development set, and use the Rich Transcription 2003 Spring evaluation CTS test set (rt03) as test set. The rt02 set consists of 6081 sentences (63804 words) and has three subsets: Switchboard 1, Switchboard 2, Switchboard Cellular. The rt03 set consists of 9050 sentences (76083 words) and has two subsets: Switchboard and Fisher.

The training set consists of 297580 transcribed utterances (3297579 words)[4]. For each utterance, a weighted word-lattice was produced, representing alternative transcriptions, from the ASR system. The baseline ASR system that we are comparing against then performed a rescoring pass on these first pass lattices, allowing for better silence modeling, and replaces the trigram language model score with a 6-gram model. 1000-best lists were then extracted from these lattices. For each candidate in the 1000-best lists, we identified the number of edits (insertions, deletions or substitutions) for that candidate, relative to the "target" transcribed utterance. The oracle score for the 1000-best lists was 16.7%.

To produce the word-lattices, each training utterance was processed by the baseline ASR system. In a naive approach, we would simply train the baseline system (i.e., an acoustic model and language model) on the entire training set, and then decode the training utterances with this system to produce lattices. We would then use these lattices with the perceptron algorithm. Unfortunately, this approach is likely to produce a set of training lattices that are very different from test lattices, in that they will have very low word-error rates, given that the lattice for each utterance was produced by a model that was trained on that utterance. To somewhat control for this, the training set was partitioned into 28 sets, and baseline Katz backoff trigram models were built for each set by including only transcripts from the other 27 sets. Lattices for each utterance were produced with an acoustic model that had been trained on the entire training set, but with a language model that was trained on the 27 data portions that did not include the current utterance. Since language models are generally far more prone to overtraining than standard acoustic models, this goes a long way toward making the training conditions similar to testing conditions. Similar procedures were used to train the parsing and tagging models for the training set, since the Switchboard treebank overlaps extensively with the ASR training utterances.

Table 2 presents the word-error rates on rt02 and rt03 of the baseline ASR system, 1000-best perceptron and GCLM results from Roark et al. (2005) under this condition, and our 1000-best perceptron results. Note that our n-best result, using just n-gram features, improves upon the perceptron result of (Roark et al., 2005) by 0.2 percent, putting us within 0.1 percent of their GCLM result for that

[4]Note that Roark et al. (2004a; 2004b; 2005) used 20854 of these utterances (249774 words) as held out data. In this work we simply use the rt02 test set as held out and development data.

Trial	WER	
	rt02	rt03
ASR system output	37.1	36.4
Roark et al. (2005) perceptron	36.6	35.7
Roark et al. (2005) GCLM	36.3	35.4
n-gram perceptron	36.4	35.5

Table 2: Baseline word-error rates versus Roark et al. (2005)

Trial	rt02 WER
ASR system output	37.1
n-gram perceptron	36.4
n-gram + POS (1) perceptron	36.1
n-gram + POS (1,2) perceptron	36.1
n-gram + POS (1,3) perceptron	36.1

Table 3: Use of POS-tag sequence derived features

condition. (Note that the perceptron–trained n-gram features were trigrams (i.e., $n = 3$).) This is due to a larger training set being used in our experiments; we have added data that was used as held-out data in (Roark et al., 2005) to the training set that we use.

The first additional features that we experimented with were POS-tag sequence derived features. Let t_i and w_i be the POS tag and word at position i, respectively. We experimented with the following three feature definitions:

1. $(t_{i-2}t_{i-1}t_i)$, $(t_{i-1}t_i)$, (t_i), (t_iw_i)
2. $(t_{i-2}t_{i-1}w_i)$
3. $(t_{i-2}w_{i-2}t_{i-1}w_{i-1}t_iw_i)$, $(t_{i-2}t_{i-1}w_{i-1}t_iw_i)$, $(t_{i-1}w_{i-1}t_iw_i)$, $(t_{i-1}t_iw_i)$

Table 3 summarizes the results of these trials on the held out set. Using the simple features (number 1 above) yielded an improvement beyond just n-grams, but additional, more complicated features failed to yield additional improvements.

Next, we considered features derived from shallow parsing sequences. Given the results from the POS-tag sequence derived features, for any given sequence, we simply use n-tag and tag/word features (number 1 above). The first sequence type from which we extracted features was the shallow parse tag sequence (S1), as shown in figure 3(b). Next, we tried the composite shallow/POS tag sequence (S2), as in figure 3(c). Finally, we tried extracting features from the shallow constituent sequence (S3), as shown in figure 3(d). When EDITED and INTJ nodes are ignored, we refer to this condition as S3-E. For full-parse feature extraction, we tried context-free rule features (CF) and head-to-head features (H2H), of the kind shown in table 1. Table 4 shows the results of these trials on rt02.

Trial	rt02 WER
ASR system output	37.1
n-gram perceptron	36.4
n-gram + POS perceptron	36.1
n-gram + POS + S1 perceptron	36.1
n-gram + POS + S2 perceptron	36.0
n-gram + POS + S3 perceptron	36.0
n-gram + POS + S3-E perceptron	36.0
n-gram + POS + CF perceptron	36.1
n-gram + POS + H2H perceptron	36.0

Table 4: Use of shallow parse sequence and full parse derived features

Although the single digit precision in the table does not show it, the H2H trial, using features extracted from the full parses along with n-grams and POS-tag sequence features, was the best performing model on the held out data, so we selected it for application to the rt03 test data. This yielded 35.2% WER, a reduction of 0.3% absolute over what was achieved with just n-grams, which is significant at $p < 0.001$,[5] reaching a total reduction of 1.2% over the baseline recognizer.

5 Conclusion

The results presented in this paper are a first step in examining the potential utility of syntactic features for discriminative language modeling for speech recognition. We tried two possible sets of features derived from the full annotation, as well as a variety of possible feature sets derived from shallow parse and POS tag sequences, the best of which gave a small but significant improvement beyond what was provided by the n-gram features. Future work will include a further investigation of parser–derived features. In addition, we plan to explore the alternative parameter estimation methods described in (Roark et al., 2004a; Roark et al., 2004b), which were shown in this previous work to give further improvements over the perceptron.

[5]We use the Matched Pair Sentence Segment test for WER, a standard measure of signifi cance, to calculate this p-value.

References

Eugene Charniak. 2001. Immediate-head parsing for language models. In *Proc. ACL*.

Ciprian Chelba and Frederick Jelinek. 1998. Exploiting syntactic structure for language modeling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 225–231.

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

Ciprian Chelba. 2000. *Exploiting Syntactic Structure for Natural Language Modeling*. Ph.D. thesis, The Johns Hopkins University.

Stanley Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report, TR-10-98, Harvard University.

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

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*, pages 1–8.

Michael Collins. 2004. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In Harry Bunt, John Carroll, and Giorgio Satta, editors, *New Developments in Parsing Technology*. Kluwer Academic Publishers, Dordrecht.

Frederick Jelinek and John Lafferty. 1991. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–323.

Mark Johnson, Stuart Geman, Steven Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proc. ACL*, pages 535–541.

Daniel Jurafsky, Chuck Wooters, Jonathan Segal, Andreas Stolcke, Eric Fosler, Gary Tajchman, and Nelson Morgan. 1995. Using a stochastic context-free grammar as a language model for speech recognition. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*, pages 189–192.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, pages 282–289, Williams College, Williamstown, MA, USA.

Andrej Ljolje, Enrico Bocchieri, Michael Riley, Brian Roark, Murat Saraclar, and Izhak Shafran. 2003. The AT&T 1xRT CTS system. In *Rich Transcription Workshop*.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of HLT-NAACL 2004*.

Brian Roark, Murat Saraclar, and Michael Collins. 2004a. Corrective language modeling for large vocabulary ASR with the perceptron algorithm. In *Proc. ICASSP*, pages 749–752.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. 2004b. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proc. ACL*.

Brian Roark, Murat Saraclar, and Michael Collins. 2005. Discriminative n-gram language modeling. *Computer Speech and Language*. *submitted*.

Brian Roark. 2001a. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2):249–276.

Brian Roark. 2001b. *Robust Probabilistic Predictive Syntactic Processing*. Ph.D. thesis, Brown University. http://arXiv.org/abs/cs/0105019.

Ronald Rosenfeld, Stanley Chen, and Xiaojin Zhu. 2001. Whole-sentence exponential language models: a vehicle for linguistic-statistical integration. In *Computer Speech and Language*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the Human Language Technology Conference and Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, Edmonton, Canada.

Andreas Stolcke and Jonathan Segal. 1994. Precise n-gram probabilities from stochastic context-free grammars. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 74–79.

Andreas Stolcke. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–202.

Wen Wang and Mary P. Harper. 2002. The superARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proc. EMNLP*, pages 238–247.

Wen Wang, Andreas Stolcke, and Mary P. Harper. 2004. The use of a linguistically motivated language model in conversational speech recognition. In *Proc. ICASSP*.

Wen Wang. 2003. *Statistical parsing and language modeling based on constraint dependency grammar*. Ph.D. thesis, Purdue University.

Peng Xu, Ciprian Chelba, and Frederick Jelinek. 2002. A study on richer syntactic dependencies for structured language modeling. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 191–198.

Peng Xu, Ahmad Emami, and Frederick Jelinek. 2003. Training connectionist models for the structured language model. In *Proc. EMNLP*, pages 160–167.