Improved Constituent Context Model with Features

Yun Huang[1,2]
huangyun@comp.nus.edu.sg

Min Zhang[1]
mzhang@i2r.a-star.edu.sg

Chew Lim Tan[2]
tancl@comp.nus.edu.sg

[1]Human Language Department
Institute for Infocomm Research
1 Fusionopolis Way, Singapore

[2]Department of Computer Science
National University of Singapore
13 Computing Drive, Singapore

Abstract

The Constituent-Context Model (CCM) achieves promising results for unsupervised grammar induction. However, its performance drops for longer sentences. In this paper, we describe a general feature-based model for CCM, in which linguistic knowledge can be easily integrated as features. Features take the log-linear form with local normalization, so the Expectation-Maximization (EM) algorithm is still applicable to estimate model parameters. The ℓ_1-norm is used to control the model complexity, leading to sparse and compact grammar. We also propose to use a separated development to perform model selection and an additional test set to evaluate the performance. Under this framework, we could automatically choose suitable model parameters rather than setting them empirically. Experiments on the English treebank demonstrate that the feature-based model achieves comparable performance on short sentences but significant improvement on longer sentences.

1 Introduction

Unsupervised grammar induction, the task to induce hierarchical structures from plain strings, has attracted research interests for a long time. The induced grammars can be used to construct large treebanks (van Zaanen, 2000), study language acquisition (Jones et al., 2010), improve machine translation (DeNero and Uszkoreit, 2011), and so on. In general, most approaches either induce the constituency grammars (Klein and Manning, 2002; Bod, 2006; Seginer, 2007; Cohn et al., 2009; Ponvert et al., 2011), or the dependency grammars (Klein and Manning, 2004; Headden III et al., 2009; Cohen and Smith, 2009; Spitkovsky et al., 2010; Blunsom and Cohn, 2010).

Among these approaches, the Constituent Context Model (CCM) (Klein and Manning, 2002; Klein, 2005) is a simple but effective generative model for unsupervised constituency grammar induction. Specifically, the sequences (the contents enclosed by spans) and contexts (the preceding and following words) are directly modelled in CCM. The Expectation-Maximization (EM) algorithm is used to estimate parameters to optimize the data likelihood. Although the CCM achieves promising results on short sentences, its performance drops for longer sentences. There are two possible reasons: (1) CCM models all constituents under only single multinomial distributions, which can not capture the detailed information of span contents; and (2) long sequences only occur a few times in the training corpus, so the probability estimation highly depends on smoothing. Another problem of original CCM and following improved unsupervised models (Smith and Eisner, 2004; Mirroshandel and Ghassem-Sani, 2008; Golland et al., 2012) is the problematic evaluation framework. The previous approaches train and evaluate models on the same dataset, so there is no reasonable way to choose model parameters unless setting them empirically.

In this paper, we focus on CCM and present a general feature-based framework in which various overlapping features could be easily added. Previous dependency induction approach (Cohen and

26th Pacific Asia Conference on Language,Information and Computation pages 564–573

Smith, 2009) demonstrates enabling factored covariance between the probabilities of different derivation events could improve the induction results. The proposed feature-based model provides a simpler and more flexible way to share information between constituents, e.g. different sequences may share the same boundary words. Various features could capture rich information about span contents, which alleviates the data sparsity problem and estimation problem of CCM mentioned above. In addition, features are combined in the log-linear form with local normalization, so the EM algorithm can be adopted to estimate model parameters with minor change, without increasing the computing complexity. To avoid overfitting, we use ℓ_1-norm regularization to control the model complexity. Finally, we advocate to estimate model probabilities on training set, use a separated development set (a.k.a. the validation set) to perform model selection, and measure the generative ability of trained model on an additional test set. Under this framework, we could automatically select suitable model and parameters rather than choosing them manually. We carry out experiments on the English treebank. Compared to original CCM, the proposed feature-based model achieves comparable performance on short sentences but significant improvement on longer sentences. After examining the effect of grammar sparsity, we conclude that with good regularization parameter (tunned on the development set), the learned grammar could be both compact and accurate.

The main contributions of this paper can be summarized as follows:

(1) We present a general feature-based CCM, where knowledge can be easily incorporated.

(2) We use ℓ_1-norm to control the model complexity, leading to compact grammars.

(3) We propose to use separated development set to tune parameters instead of heuristically choosing parameters.

This paper is structured as follows. Section 2 gives an overview of the original CCM. Section 3 proposes the feature-based CCM and corresponding parameter estimation method. Section 4 lists the feature templates used in experiments. Section 5 shows the experimental results. We compare our work to related approaches in Section 6 and conclude in Section 7.

2 Constituent Context Model

The Constituent-Context Model (CCM) (Klein and Manning, 2002) is the first model achieving better performance than the trivial right branching baseline in the unsupervised English grammar induction task. Unlike many models that only deal with constituent spans, the CCM defines generative probabilistic models over all spans of a sentence, no matter whether they enclose constituents or distituents (a.k.a. the non-constituents).

In particular, let B be a boolean matrix with entries indicating whether the corresponding span encloses constituent or distituent. Note that each tree could be represented by one and only one bracketing, but some bracketings are not tree-equivalent, since they may miss the full-sentence span or have crossing spans. Define sequence σ to be the substring enclosed by span, and context γ to be the pair of preceding and following terminals[1]. The CCM generate a sentence S in two steps: first choose a bracketing B according to prior distribution, then generate the sentence given the chosen bracketing:

$$P(S,B) = P(B)P(S|B).$$

The prior $P(B)$ uniformly distributes its probability mass over all possible binary trees of the given sentence, and zero for non-tree-equivalent bracketings. The conditional probability $P(S|B)$ is further decomposed to the product of generative probability of sequence σ and context γ for each span $\langle i,j\rangle$:

$$\begin{aligned} P(S|B) &= \prod_{\langle i,j\rangle} P(\sigma_{\langle i,j\rangle}, \gamma_{\langle i,j\rangle} | B_{\langle i,j\rangle}) \\ &= \prod_{\langle i,j\rangle} P(\sigma_{\langle i,j\rangle} | B_{\langle i,j\rangle}) P(\gamma_{\langle i,j\rangle} | B_{\langle i,j\rangle}). \quad (1) \end{aligned}$$

From the above decomposition, we can see that given B, the CCM fills each span independently and generates yield and context independently.

The Expectation Maximization (EM) algorithm is used to estimate the multinomial parameters $\boldsymbol{\theta}$. In the E-step, a cubic-time dynamic programming algorithm is used to calculate the expected counts for

[1] For example, in sequence "${}_0\texttt{RB}_1\texttt{DT}_2\texttt{NN}_3$", we have $\sigma_{\langle 1,3\rangle} = \langle \texttt{DT NN}\rangle$, and $\gamma_{\langle 1,3\rangle} = \langle \texttt{RB}, \diamond\rangle$. Since CCM works on part-of-speech (POS) tags, only POS tags are shown here. The special symbol $\diamond$ represents the sentence boundary.

each sequence and context for both constituents and distituents according to the current $\boldsymbol{\theta}$. The detailed calculation of expectation can be found in Appendix A.1 in (Klein, 2005). In the M-Step, the model finds new $\boldsymbol{\theta}'$ to maximize the expected completed likelihood $\sum_B P(B|S, \boldsymbol{\theta}^{old}) \log P(S, B|\boldsymbol{\theta}')$ by normalizing relative frequencies.

From the probability definition (1), the CCM gives single multinomial probability distribution over all sequences. However, the number of possible sequences grows exponentially with respect to the span length, leading to severe data sparsity problem for long sentences. In the next section, we propose the feature-based model to alleviate the this problem, since overlapping features could represent small units of the span contents.

3 Feature-based CCM

3.1 Model Definition

Motivated by (Berg-Kirkpatrick et al., 2010), we define factors in the log-linear form with local normalization. Let $F_{1,\ldots,K}$ be K different factors. Each factor F_k corresponds to a n_k-dimensional feature vector $\boldsymbol{f}_k$ and a n_k-dimensional weight vector $\boldsymbol{w}_k$. For the k^{th} factor F_k, the corresponding multinomial parameter in original CCM is now treated as a function of weights $\boldsymbol{w}_k$. Define the factor category function δ_k to be $+1$ if F_k is constituent factor, and -1 otherwise. In detail, for span $\langle i, j\rangle$ in some bracketing B for sentence S, define

$$\begin{aligned} F_k(S_{\langle i,j\rangle}|\boldsymbol{w}_k) &= P_k(S_{\langle i,j\rangle}|B_{\langle i,j\rangle} = \delta_k, \boldsymbol{w}_k) \\ &= \frac{\exp(\boldsymbol{w}_k \cdot \boldsymbol{f}_k(S_{\langle i,j\rangle}))}{\sum_v \exp(\boldsymbol{w}_k \cdot \boldsymbol{f}_k(v))} \end{aligned} \quad (2)$$

where $\boldsymbol{f}_k$ returns a feature vector, $\boldsymbol{w}_k$ is the corresponding weight vector, and $(\cdot)$ denotes the inner product of vectors. The denominator sums over the unnormalized probabilities (the numerator) for all possible factor values v. We approximately calculate this summation only over values that appear in training corpus.

For factor F_k over bracketing B with corresponding tree T_B, define the active span set $\mathcal{A}_k(B)$ as

$$\mathcal{A}_k(B) = \begin{cases} \{\langle i,j\rangle \in T_B\}, & \text{if } \delta_k = +1 \\ \{\langle i,j\rangle \notin T_B\}, & \text{if } \delta_k = -1 \end{cases} \quad (3)$$

Then the joint probability of $P(S, B|\boldsymbol{w})$ can be defined:

$$\begin{aligned} P(S, B|\boldsymbol{w}) &= P(B)P(S|B) \\ &= P(B) \prod_{\langle i,j\rangle} P(S_{\langle i,j\rangle}|B_{\langle i,j\rangle}) \\ &= P(B) \prod_{\langle i,j\rangle \in \mathcal{A}_k(B)} F_k(S_{\langle i,j\rangle}|\boldsymbol{w}_k) \\ &= P(B) \prod_{\langle i,j\rangle} \prod_{k:\delta_k=-1} F_k(S_{\langle i,j\rangle}|\boldsymbol{w}_k) \\ &\quad \times \prod_{\langle i,j\rangle \in T_B} \frac{\prod_{k:\delta_k=1} F_k(S_{\langle i,j\rangle}|\boldsymbol{w}_k)}{\prod_{k:\delta_k=-1} F_k(S_{\langle i,j\rangle}|\boldsymbol{w}_k)} \\ &= K(S|\boldsymbol{w}) \prod_{\langle i,j\rangle \in T_B} \prod_k F_k^{\delta_k}(S_{\langle i,j\rangle}|\boldsymbol{w}_k) \end{aligned}$$

where $K(S|\boldsymbol{w})$ is independent of B and the following production is taken over tree spans only. One advantage of the locally normalized model is that the EM algorithm could be still used to estimate parameters, which will be described in the next subsection.

If we define the same factors of CCM (sequence and context for constituent and distituent) and set weights properly, then the probability of feature-based model is degenerated to the original CCM model. So the original CCM can be treated as a special case of the feature-based model.

3.2 Parameter Estimation

Let $\mathcal{S}$ be the set of training sentences. Under the maximum likelihood estimation, we want to find $\boldsymbol{w}$ to maximize the data log likelihood:

$$L(\mathcal{S}|\boldsymbol{w}) = \sum_{S \in \mathcal{S}} \log \sum_{B \in \mathcal{B}(S)} P(S, B|\boldsymbol{w}) \quad (4)$$

However, the summation of hidden variable B is inside the logarithm operator, resulting in the complicated expressions for the analytical solution. Instead, we use the Expectation-Maximization (EM) algorithm to solve the problem approximately.

Given current model parameters $\boldsymbol{w}^{old}$ in each iteration of EM, we seek new parameter $\boldsymbol{w}$ to maximize the expectation of the completed-data log likelihood:

$$\begin{aligned} &Q(\boldsymbol{w}, \boldsymbol{w}^{old}) \\ &= \sum_{S \in \mathcal{S}} \sum_{B \in \mathcal{B}(S)} P(B|S, \boldsymbol{w}^{old}) \log P(S, B|\boldsymbol{w}) \end{aligned} \quad (5)$$

E-Step

The E-step evaluates the posterior probability $P(B|S, \boldsymbol{w}^{old})$ given fixed $\boldsymbol{w}^{old}$. We modify the inside-outside algorithm (Lari and Young, 1990) to efficiently calculate the expected count for each factor. The original inside/outside merits are recursively calculated over binary rules. In the feature-based CCM, we recursively calculate these values over spans. To simplify following derivations, we define

$$\phi_{\langle i,j\rangle} = \prod_k F_k^{\delta_k}(S_{\langle i,j\rangle}|\boldsymbol{w}_k) \tag{6}$$

The inside probability $\textsc{In}_{\langle i,j\rangle}$ can be defined recursively:

(a) Unary spans: $\textsc{In}_{\langle i,j\rangle} = \phi_{\langle i,j\rangle}$, if $j - i = 1$;

(b) Other spans:

$\textsc{In}_{\langle i,j\rangle} = \sum_{k=i+1}^{j-1} \phi_{\langle i,j\rangle}\, \textsc{In}_{\langle i,k\rangle}\, \textsc{In}_{\langle k,j\rangle}$.

For sentence S with length l, the outside probability can be defined as:

(a) Sentence span: $\textsc{Out}_{\langle 0,l\rangle} = 1$;

(b) Other spans:

$$\textsc{Out}_{\langle i,j\rangle} = \sum_{k=0}^{i-1} \phi_{\langle k,j\rangle}\, \textsc{Out}_{\langle k,j\rangle}\, \textsc{In}_{\langle k,i\rangle} + \sum_{k=j+1}^{l} \phi_{\langle i,k\rangle}\, \textsc{Out}_{\langle i,k\rangle}\, \textsc{In}_{\langle j,k\rangle}.$$

Then we calculate the expected ratio $\phi_{\langle i,j\rangle}$ for each span:

$$e[\phi_{\langle i,j\rangle}] = \textsc{In}_{\langle i,j\rangle} \times \textsc{Out}_{\langle i,j\rangle} \,/\, \textsc{In}_{\langle 0,l\rangle} \tag{7}$$

Finally, we accumulate expected counts e and $1 - e$ constituent factors and distituent factors respectively.

We do not consider empty spans in the above calculation of inside/outside probabilities. Since the empty spans do not depend on trees, we just add expected count 1 for each distituent factor and 0 for each constituent factor over empty spans.

M-Step

In M-step, we want to tune $\boldsymbol{w}$ to maximize the expected complicated log likelihood together with the regularization terms:

$$Q(\boldsymbol{w}, \boldsymbol{w}^{old}) - \sum_{k=1}^{K} \lambda_k \|\boldsymbol{w}_k\|_1 \tag{8}$$

where λ_k is a non-negative coefficient for the ℓ_1-norm of the k^{th} weight vector $\boldsymbol{w}_k$. Because of the high-dimensional feature space, we use ℓ_1-norm of weight vector $\boldsymbol{w}$ as regularization terms to control the model complexity. The regularization terms can serve as automatic feature selector, leading to compact models.

In original CCM, model parameters (multinomial distribution probabilities) are estimated by normalizing relative frequencies in the M-step. In the feature-based model, we use gradient-based search algorithm to optimize the above objective function numerically. Due to the ℓ_1-norm regularization, the objective is not differentiable at $\boldsymbol{w} = \boldsymbol{0}$. So we adopt the OWL-QN method (Andrew and Gao, 2007) to perform optimization. The open-source C++ implementation `libLBFGS`[2] is used in experiments. The optimization process needs to calculate the gradient of $Q(\boldsymbol{w}, \boldsymbol{w}^{old})$ with respect to $\boldsymbol{w}$.

Since the probabilities of factors are multiplied together, so the logarithm term in equation (5) can be decomposed into the sum of the logarithm of each factor probability. Additionally, the ℓ_1-norm term in equation (8) is the sum of ℓ_1-norm of the weights for each factor. As a result, optimizing the overall objective function is equivalent to optimize the corresponding functions for each factor.

Assuming the set $\mathcal{V}_k$ contains all values of the k^{th} factor F_k that can be found in training corpus, then the gradient (omitting the regularization terms) of Q_k can be computed as follows:

$$\boldsymbol{\nabla}_{\boldsymbol{w}_k}(Q_k) = \sum_{v \in \mathcal{V}_k} e[F_k(v)] \times \boldsymbol{\Delta}_v(\boldsymbol{w}_k) \tag{9}$$

$$\boldsymbol{\Delta}_v(\boldsymbol{w}_k) = \boldsymbol{f}_k(v) - \sum_{v' \in \mathcal{V}_k} F_k(v')\boldsymbol{f}_k(v') \tag{10}$$

where $e[F_k(v)]$ contains the expected counts calculated in the E-step. The similar derivation can be found in (Berg-Kirkpatrick et al., 2010).

In this feature-based model, rich features can be easily incorporated. We give some useful feature templates in next section.

[2] `http://www.chokkan.org/software/liblbfgs/`

4 Feature Templates

4.1 Basic features

There are two kinds of features: constituent features, with prefix {c:}; and distituent features, with prefix {d:}. Features in the two categories are active only if the span enclose constituent or distituent respectively. The basic feature templates are listed as follows with their names and descriptions. A running example, span $\langle 1, 3\rangle$ in "$_0$RB$_1$DT$_2$NN$_3$", is also shown for each feature template.

• const: This constant feature always takes value 1 for any given span. We use this feature to measure the number of spans.

• seq[n]: This indicating feature is active for sequence enclosed by span with size n. If $n = 0$, then sequences with any lengths are considered.

seq2	...	DT_JJ	DT_NN	RB_DT	...
value	...	0	1	0	...

• lx[n]/rx[n]: The indicating feature for the preceding/following n terminals (left/right context), where ◇ represents sentence boundary.

lx2	...	◇_◇	◇_RB	RB_DT	...
value	...	0	1	0	...

rx2	...	DT_NN	NN_◇	◇_◇	...
value	...	0	0	1	...

• lb[n]/rb[n]: The left/right n boundary terminals inside given span. If the length of span is less than n, then this feature template is not activated.

lb2	...	RB_DT	DT_NN	TO_VB	...
value	...	0	1	0	...

rb1	...	RB	DT	NN	...
value	...	0	0	1	...

4.2 Composite features

Basic features can be composited to more complicated features. We define two composition operators: join (.), and concatenation (+). For the join operator, the composited feature space is the Cartesian product of the feature spaces of the two operands. For the concatenation operator, the composited feature space is the concatenation of the operands' feature spaces.

Here we use an example to demonstrate the difference between join operator and concatenation operator. Assuming there are three possible values {◇, RB, DT} for feature lx1, and three possible values {DT, NN, ◇} for feature rx1, then the joined feature space has $3 \times 3 = 9$ dimensions while the concatenated feature space has $3 + 3 = 6$ dimensions. The feature vectors of these two operators for span $\langle 1, 3\rangle$ in "$_0$RB$_1$DT$_2$NN$_3$" are shown as follows.

lx1.rx1								
◇.{DT,NN,◇}			RB.{DT,NN,◇}			DT.{DT,NN,◇}		
0	0	0	0	0	1	0	0	0

lx1 + rx1					
◇	RB	DT	DT	NN	◇
0	1	0	0	0	1

We restrict that only join followed by concatenation is allowed. As an example, the original CCM could be represented as: {c:seq0, d:seq0, c:lx1.rx1, d:lx1.rx1}.

4.3 Summary

There are huge number of feature combinations that we can not try each of them in experiments. In experiments, we use following sets of features.

The first feature set includes the sequences with length up to 5: {seq1, seq2, seq3, seq4, seq5}. Note that sequences with arbitrary lengths are modelled in the original CCM, while we restrict the maximal sequence length to be 5. Since most of the longer sequences occurs only once or twice in the training corpus, we discard them to speed up training procedure and reduce memory usage.

Boundary words have been proven useful for detecting phrase boundaries in supervised learning task (Xiong et al., 2010). We introduce this idea to unsupervised grammar induction. The features used in experiments are combinations of left boundary and right boundary words with lengths up to 2: {lb1, lb2, rb1, rb2, lb1.rb1, lb1.rb2, lb2.rb1, lb2.rb2}.

The original CCM also considers the pair of preceding one word and following one word as contexts. We consider combinations of left context and right context words with lengths up to 2: {lx1, lx2, rx1, rx2, lx1.rx1, lx1.rx2, lx2.rx1, lx2.rx2}. The special token ◇ is introduced to represent sentence boundaries.

The last feature used is the constant feature {const}. The constant feature always takes value 1 for each span.

Overall, we define two constituent factors and two distituent factors. The first constituent/distituent factors, denoted as $F_{\texttt{c:s}}$ and $F_{\texttt{d:s}}$, are the concatenation of sequence features, boundary features, and constant feature. The second constituent/distituent factors ($F_{\texttt{c:x}}$ and $F_{\texttt{d:x}}$), are the concatenation of context features and constant feature.

5 Experiments

5.1 Datasets and Settings

We carry out experiments on the Wall Street Journal portion of the Penn English Treebank (Marcus et al., 1993). We report the unlabeled F1 score (the harmonic mean of precision and recall) as evaluation metric. Constituents which could not be gotten wrong (single words and entire sentences) are discarded. These are standard settings used in previous work (Klein, 2005).

To perform model selection and parameter tunning, we split the treebank into three parts: section 02-21 as training set, section 00 as development set, and section 23 as test set. As standard machine learning pipeline, we perform EM on training set, tune parameters on development set, and report the result of selected model on test set. We remove punctuation and null elements in treebank, as the standard preprocessing step (Klein, 2005). For comparison, we build various datasets with sentences lengths no more than 10, 20, 30, 40 words after removing punctuations. Table 1 gives the number of sentences for each dataset.

Dataset	Train	Dev	Test
PTB10	5,899	265	398
PTB20	20,243	992	1,286
PTB30	32,712	1,573	2,028
PTB40	37,561	1,809	2,338

Table 1: Data statistics

We select regularization parameters from set $\{0.03, 0.1, 0.3, 1, 3, 10\}$ for factors $F_{\texttt{c:s}}$ and $F_{\texttt{d:s}}$. No regularization is used for factor $F_{\texttt{c:x}}$ and $F_{\texttt{d:x}}$, since the number of context types are almost fixed and relatively small in datasets with different lengths. Each combinations of $\lambda_{\texttt{c:s}}$ and $\lambda_{\texttt{d:s}}$ are tested on the development set. The final values of λ is the one with the highest development F1 score.

5.2 Induction Results

EM algorithm is sensitive to the initial condition. We adopt the same uniform-split initialization and the same smoothing values (2 for constituents and 8 for distituents) as described in (Klein, 2005). For feature-based model (F-CCM), we still use uniform-split strategy to initialize probabilities in the first E-step, and set all weights to zero as the initial point of the gradient-based search algorithm in the M-step.

PTB10	Train	Dev	Test
LBranch	28.62	28.64	30.58
RBranch	61.58	63.59	61.00
UBound	88.20	88.35	86.80
CCM	**72.50**	**73.58**	**70.30**
F-CCM	71.66	72.95	69.75
PTB20	Train	Dev	Test
LBranch	17.22	17.43	17.21
RBranch	48.39	47.85	47.96
UBound	86.35	86.26	86.20
CCM	48.96	48.46	48.08
F-CCM	**59.86**	**59.86**	**59.10**
PTB30	Train	Dev	Test
LBranch	13.37	13.61	13.33
RBranch	42.70	42.76	42.57
UBound	85.72	86.02	85.88
CCM	43.01	43.27	42.59
F-CCM	**48.87**	**48.82**	**48.15**
PTB40	Train	Dev	Test
LBranch	12.08	12.31	11.95
RBranch	40.59	40.54	40.73
UBound	85.54	85.77	85.69
CCM	33.44	33.62	33.10
F-CCM	**45.44**	**45.46**	**45.10**

Table 2: Results on PTB10, PTB20, PTB30, PTB40

Table 2 shows the experimental results on the datasets of different length limits. LBranch and RBranch rows show the left branching and right branching binary tree baselines. As the English grammar tends to be right branched, the trivial RBranch achieves quite high F1 scores. UBound rows show the results of binarized treebank, which is the upper bound of any grammar induction systems that output binary trees. We reimplement the baseline CCM, which achieves comparable performance compared to previous reported results (Klein, 2005). The results of feature-based CCM are presented in the F-CCM rows.

From these results, we observe that the original CCM performs much better than the right branching baseline on short sentences, but the performance decreases dramatically on longer sentences, even lower than the right branching baseline. In contrast, our feature-based CCM achieves comparable performance with original CCM on PTB10, and much better performance than the original CCM and the right branching baseline on longer sentences. These experimental results demonstrate the effectiveness and robustness of the feature-based model.

5.3 Grammar sparsity

The regularization terms can serve as feature selection mechanism. In this section, we compare the sparsity of learned sequence grammars between various regularization coefficients on PTB10.

$\lambda_{\mathtt{c:s}}$	$\lambda_{\mathtt{d:s}}$	$F_{\mathtt{c:s}}$	$F_{\mathtt{d:s}}$	Dev
0	0	72,289	72,289	69.56
0.1	0.03	55,407	71,668	70.39
0.1	0.1	54,591	69,988	70.87
0.1	0.3	56,660	57,729	70.32
0.1	1	55,860	27,058	**72.95**
0.1	3	55,534	9,885	60.82
0.1	10	56,513	3,149	55.55
0.03	1	69,390	28,046	67.13
0.1	1	55,860	27,058	**72.95**
0.3	1	31,763	27,525	70.32
1	1	11,816	27,418	71.01
3	1	4,040	27,559	71.08
10	1	1,456	27,875	72.26

Table 3: Number of non-zero weights for factors $F_{\mathtt{c:s}}$ and $F_{\mathtt{d:s}}$. The corresponding F1 scores on the development set are shown in the last column.

As mentioned in section 5.1, we only tune regularization parameters $\lambda_{\mathtt{c:s}}$ and $\lambda_{\mathtt{d:s}}$. We can not report results of all combinations since there are too many of them. Instead, we report results with either $\lambda_{\mathtt{c:s}}$ or $\lambda_{\mathtt{d:s}}$ fixed to the best tunned value. The number of active dimensions (i.e. with non-zero weight) and the development F1 score are examined in experiments.

Table 3 shows the results of these experiments. With the increasing of regularization parameters, the model becomes more and more sparser (as measured by the number of non-zero weights). The tuned optimal parameter values are $\lambda_{\mathtt{c:s}} = 0.1$ and $\lambda_{\mathtt{d:s}} = 1$. It is interesting to observe that the optimal regularization value for distituent factor is greater than the one for constituent factor. This fact can be explained that since there are more distituents than constituents, the distituent weights need to be penalized more heavily. The best development F1 is 72.95, greater than the F1 score 69.56 achieved without regularization, since the unregularized model may overfit the training data.

If we compare experiments with fixed $\lambda_{\mathtt{c:s}}$, the development F1 score first increases and then decreases with the increase of $\lambda_{\mathtt{d:s}}$. In contrast, with fixed $\lambda_{\mathtt{d:s}}$, the performance varies little for different $\lambda_{\mathtt{c:s}}$. These results somehow demonstrate the distituents modelled in CCM play a more important role than the constituents.

5.4 Discussion

Experiments show that we achieve better performance than original CCM while using compact grammars. There are some issues we want to discuss here.

1. We only test a few feature templates. Other features such as words, stems may improve the results. Moreover, punctuations contain useful information in grammar induction (Spitkovsky et al., 2011b; Ponvert et al., 2011), while currently punctuations are ignored in our model.
2. In previous unsupervised constituency grammar induction, how to choose parameters is an art. While in the proposed model, we use development set to perform model selection.
3. EM algorithm could only find sub-optima. One possible solution is the Lateen EM (Spitkovsky et al., 2011a), in which multiple objective functions are an alternative optimized. Another method is the annealing technique during probability estimation process. We will investigate these in future work.
4. ℓ_1-norm regularization is used to learn sparse and compact model. Bayesian learning is an alternative framework, which can be also applied to CCM.

6 Related Work

The Constituent-Context Model (Klein and Manning, 2002; Klein, 2005) is the first unsupervised constituency grammar induction system that achieves better performance than the trivial right branching baseline for English. However, the performance of CCM degrades on long sentences. Following approaches improve CCM in various aspects. Smith and Eisner (2004) propose to condition the yield feature on the length of the yield and use annealing techniques to estimate parameters. Annealing techniques can be also used for the proposed F-CCM, which we plan to do in future work. Klein and Manning (2004) demonstrate the joint model of constituency and dependency could improve unsupervised grammar inference. Some other approaches also consider to use additional information such as the words (Headden III et al., 2009), the automatic induced tags (Headden III et al., 2008), or the parent information (Mirroshandel and Ghassem-Sani, 2008). These information could be easily incorporated to the proposed model as features.

Feature-based models have been widely used in many supervised tasks such as parsing (Charniak, 2000), word alignment (Moore, 2005; Liu et al., 2006), machine translation (Koehn et al., 2003), etc. For the unsupervised learning tasks, the calculation of normalization part is usually time-consuming or even impossible. Existing approaches are mainly based on the contrastive estimation (Smith and Eisner, 2005; Smith and Eisner, 2005; Dyer et al., 2011) to learn parameters. The local normalized feature-based model has been proposed in (Berg-Kirkpatrick et al., 2010), in which features are defined over generative rules and the normalization is done locally. They use the ℓ_2 regularization, while we apply the local-normalization model to CCM with ℓ_1 regularization and show improved performance could be achieved with sparse solution. Many unsupervised approaches aim to learn compact and sparse grammar, including the Bayesian models (Johnson et al., 2007; Cohn et al., 2010; Blunsom and Cohn, 2010) and posterior regularization (Ganchev et al., 2010). We use alternative (and simpler) regularization technique to obtain sparse solution.

The most related work is (Golland et al., 2012), in which a similar feature-based model for CCM is proposed. There are many differences between their work and our proposed model. (1) We use ℓ_1 regularization to learn sparse model, while they do not mention sparsity problem. (2) We propose to use a separated development set to perform model selection and an additional test set to report final results, while they directly train and evaluate their model on the same dataset, which is problematic. (3) We evaluate different feature sets from theirs. The limited lengths for sequences could reduce the memory usage for long sentences.

7 Conclusion

The constituent-context model performs well on short sentences, but the performance degrades on longer sentences. We present a feature-based model for CCM, in which linguistic knowledge can be integrated as features. Features take the log-linear form with local normalization, so the EM algorithm is still applicable to estimate model parameters. To avoid overfitting, we use the ℓ_1-norm regularization to control model complexity. We also proposed a reasonable model selection and evaluation framework. Experimental results demonstrate that the feature-based model achieves comparable performance on short sentences but significantly outperforms the original CCM on longer sentences.

Acknowledgments

We would like to thank Zhonghua Li for insightful discussion about the feature-based model and help on the numeric optimization toolkit. We also thank Xiangyu Duan for his help on the reimplementation of original CCM. Thank the anonymous reviewers for their helpful comments and suggestions.

References

Galen Andrew and Jianfeng Gao. 2007. Scalable training of l1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590.

Phil Blunsom and Trevor Cohn. 2010. Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1204–1213.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 865–872.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139.

Shay Cohen and Noah A. Smith. 2009. Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82.

Trevor Cohn, Sharon Goldwater, and Phil Blunsom. 2009. Inducing compact but accurate tree-substitution grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 548–556.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2010. Inducing Tree-Substitution grammars. *Journal of Machine Learning Research*, 11:3053–3096.

John DeNero and Jakob Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 193–203.

Chris Dyer, Jonathan H. Clark, Alon Lavie, and Noah A. Smith. 2011. Unsupervised word alignment with arbitrary features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 409–419.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

Dave Golland, John DeNero, and Jakob Uszkoreit. 2012. A feature-rich constituent context model for grammar induction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 17–22.

William P. Headden III, David McClosky, and Eugene Charniak. 2008. Evaluating unsupervised part-of-speech tagging for grammar induction. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 329–336.

William P. Headden III, Mark Johnson, and David McClosky. 2009. Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007. Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Advances in Neural Information Processing Systems 19*, pages 641–648.

Bevan K. Jones, Mark Johnson, and Michael C. Frank. 2010. Learning words and their meanings from unsegmented child-directed speech. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 501–509.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 478–485.

Dan Klein. 2005. *The Unsupervised Learning of Natural Language Structure*. Ph.D. thesis, Stanford University.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 48–54.

K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4:35–56.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Seyed Abolghasem Mirroshandel and Gholamreza Ghassem-Sani. 2008. Unsupervised grammar induction using a parent based constituent context model. In *Proceedings of the 18th European Conference on Artificial Intelligence*, pages 293–297.

Robert C. Moore. 2005. A discriminative framework for bilingual word alignment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 81–88.

Elias Ponvert, Jason Baldridge, and Katrin Erk. 2011. Simple unsupervised grammar induction from raw text with cascaded finite state models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1077–1086.

Yoav Seginer. 2007. Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 384–391.

Noah A. Smith and Jason Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 486–493.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 354–362.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011a. Lateen EM: Unsupervised training with multiple objectives, applied to dependency grammar induction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1269–1280.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2011b. Punctuation: Making a point in unsupervised dependency parsing. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 19–28.

Menno van Zaanen. 2000. ABL: Alignment-based learning. In *Proceedings of the 18th International Conference on Computational Linguistics (Coling 2000)*, volume 2, pages 961–967.

Deyi Xiong, Min Zhang, and Haizhou Li. 2010. Learning translation boundaries for phrase-based decoding. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 136–144.