Unsupervised Discriminative Induction of Synchronous Grammar for Machine Translation

Xinyan Xiao[†‡] Deyi Xiong[*] Yang Liu[⋆] Qun Liu[†] Shouxun Lin[†]
[†]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[‡]Youdao Inc., Beijing, China
[*]Human Language Technology, Institute for Infocomm Research, Singapore
[⋆]Department of Computer Sci. and Tech., Tsinghua University, Beijing, China
xiaoxy@rd.netease.com, dyxiong@i2r.a-star.edu.sg,
liuyang2011@tsinghua.edu.cn, liuqun@ict.ac.cn, sxlin@ict.ac.cn

ABSTRACT
We present a global log-linear model for synchronous grammar induction, which is capable of incorporating arbitrary features. The parameters in the model are trained in an unsupervised fashion from parallel sentences without word alignments. To make parameter training tractable, we also propose a novel and efficient cube pruning based synchronous parsing algorithm. Using learned synchronous grammar rules with millions of features that contain rule level, word level and translation boundary information, we significantly outperform a competitive hierarchical phrased-based baseline system by +1.4 BLEU on average on three NIST test sets.

KEYWORDS: synchronous grammar induction, discriminative model, unsupervised learning, machine translation.

Proceedings of COLING 2012: Technical Papers, pages 2883–2898,
COLING 2012, Mumbai, December 2012.

1 Introduction

In the last decade, statistical machine translation (SMT) has been advanced by expanding the basic unit of translation from word to phrase (Koehn et al., 2003) and grammar (Galley et al., 2006; Liu et al., 2006; Chiang, 2007). Most systems induce synchronous grammars (including phrases) from parallel corpora using a heuristic two-step pipeline. This pipeline first aligns a parallel corpus at the word level with heuristic word alignment combination strategies (e.g., grow-diag-final-and) (Koehn et al., 2003), and then extracts translation rules from word-aligned sentence pairs. It is working well in practice and therefore widely adopted. However, such a pipeline artificially brings an undesirable disconnection between translation model and word alignment model (Blunsom et al., 2009; DeNero and Klein, 2010).

Recently, researchers have resorted to principled probabilistic formulations. Various generative models are proposed to learn translation rules directly from sentence pairs without word alignments (Marcu and Wong, 2002; Cherry and Lin, 2007; Zhang et al., 2008; DeNero et al., 2008; Blunsom et al., 2009; Cohn and Blunsom, 2009; Neubig et al., 2011; Levenberg et al., 2012). Due to the independency assumptions in such generative models, it is hard to extend them to incorporate arbitrary features, especially word alignment information as used in the traditional two-step pipeline. As a result, despite theoretical advantages of these models over the two-step pipeline, in practice they can only produce comparable translation quality with the two-step pipeline in some scenarios, and usually even worse results.

Yet another alternative for synchronous grammar induction is unsupervised discriminative model. Unsupervised discriminative model can directly learn synchronous grammars in a theoretically justified manner just like generative model. However, the advantage over generative model is that it is able to easily incorporate word alignment information which has been proved useful in the two-step pipeline.

In this paper, we propose a global log-linear model (Sec. 2) for the induction of synchronous context free grammar (SCFG) (Chiang, 2007). The log-linear model is able to incorporate arbitrary features. Furthermore, it is trained from sentence pairs without word alignments in an unsupervised fashion. In particular:

- We approximate the exact conditional log-likelihood objective inspired by contrastive estimation (Smith and Eisner, 2005) as the optimization of the exact objective is very expensive. The key idea is to estimate parameters via *synchronous hypergraphs* of sentence pairs and *neighbor source hypergraphs* of source sentences (Sec. 3).
- Synchronous parsing is often impractical in large-scale learning applications due to its high complexity $O(n^6)$. We address this challenge by proposing a novel and efficient $O(n^3)$ cube pruning based synchronous parsing algorithm (Sec. 4).
- Aiming to enhance the ability to predict whether a translation derivation is good or not, we incorporate a variety of fine-grained features into our model, including rule level features, word level features and phrase boundary features (Xiong et al., 2010) (Sec. 5).

We evaluate our approach on the NIST Chinese-English translation task. According to the analysis of grammar (Sec. 6.2), our induced grammar is more reusable, and is able to generate better (+8.3 BLEU points) oracle translations than the grammar of the baseline. Meanwhile, in the end-to-end machine translation experiments, our approach outperforms the two-step pipeline by +1.4 BLEU points (Sec. 6.3).

2 Global Log-linear Model

We propose a log-linear model to induce SCFG rules for hierarchical phrase-based translation (Chiang, 2007) which transforms a source sentence $\mathbf{s}$ into a target sentence $\mathbf{t}$ by a sequence of SCFG rules. Such a sequence of rules $\{r\}$ is called a **derivation d**.

As the training data only contains sentence pairs, we model the derivation as a latent variable. The conditional probability $p(\mathbf{t}|\mathbf{s})$ of a target sentence given a source sentence is defined as the sum over all possible derivations $\mathbf{d}$:

$$p(\mathbf{t}|\mathbf{s}) = \sum_{\mathbf{d} \in \triangle(\mathbf{t},\mathbf{s})} p(\mathbf{d},\mathbf{t}|\mathbf{s}) \tag{1}$$

where $\triangle(\mathbf{t},\mathbf{s})$ is the set of all possible derivations that translate $\mathbf{s}$ into $\mathbf{t}$, and $\mathbf{d}$ is one such derivation. Given a source sentence $\mathbf{s}$, the conditional probability of a derivation $\mathbf{d}$ and the corresponding translation $\mathbf{t}$ is:

$$p(\mathbf{d},\mathbf{t}|\mathbf{s}) = \frac{\exp \sum_i \lambda_i H_i(\mathbf{d},\mathbf{t},\mathbf{s})}{Z(\mathbf{s})} \tag{2}$$

where $H_i(\mathbf{d},\mathbf{t},\mathbf{s}) = \sum_{r \in \mathbf{d}} h_i(r,\mathbf{s})$ is feature function. We assume H_i decomposes with derivation $\mathbf{d}$ in terms of local feature function h_i, which is related to a rule r and a source sentence $\mathbf{s}$. λ_i is the correspondent feature weight. $Z(\mathbf{s})$ is the partition function:

$$Z(\mathbf{s}) = \sum_{\mathbf{t}} \sum_{\mathbf{d} \in \triangle(\mathbf{t},\mathbf{s})} \exp \sum_i \lambda_i H_i(\mathbf{d},\mathbf{t},\mathbf{s}) \tag{3}$$

Such a discriminative latent variable model is not new to SMT (Blunsom et al., 2008; Kääriäinen, 2009; Xiao et al., 2011). However, we are distinguished from previous work by applying this model to synchronous grammar induction. The purpose of the latent variable model in such previous work is to do max-translation decoding and training (Blunsom et al., 2008), or to eliminate the gap between heuristic extraction and decoding (Kääriäinen, 2009), instead of grammar induction as synchronous rules are still extracted by the heuristic two-step pipeline. In contrast, our interest lies in using latent variable model to learn synchronous grammar directly from sentence pairs. Overall, to the best of our knowledge, this is the first attempt to apply this log-linear model for synchronous grammar induction.

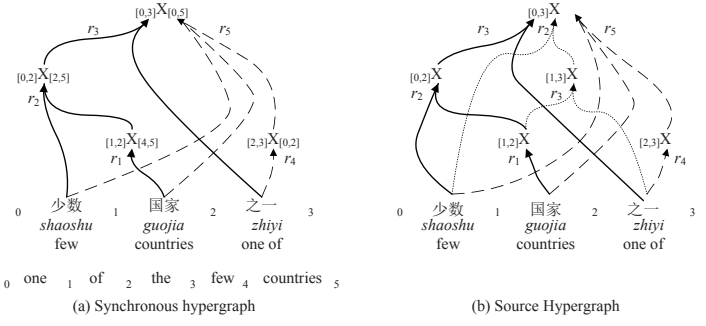
3 Training

We use maximum a posteriori estimator with stochastic gradient descent algorithm for optimization. We maximize the log-likelihood $\mathbb{L}$ of the bilingual corpus $\mathbf{T} = \{(\mathbf{s}^n, \mathbf{t}^n)\}_{n=1}^N$, penalized by a gaussian prior with mean 0 and variance σ (L_2 norm):

$$\mathbb{L} = \sum_{(\mathbf{s}^n,\mathbf{t}^n) \in \mathbf{T}} \log p(\mathbf{t}|\mathbf{s}) + \sum_i \log p_0(\lambda_i) \tag{4}$$

The gradient of the above objective is as follows:

$$\frac{\partial \mathbb{L}}{\partial \lambda_i} = E_{p(\mathbf{d}|\mathbf{t},\mathbf{s})}[H_i] - E_{p(\mathbf{d},\mathbf{t}|\mathbf{s})}[H_i] - \frac{\lambda_i}{\sigma^2} \tag{5}$$



r_1: $\langle$ *guojia* $\Rightarrow$ countries $\rangle$ r_2: $\langle$ *shaoshu* X $\Rightarrow$ the few X $\rangle$
r_3: $\langle$ X *zhiyi* $\Rightarrow$ one of X $\rangle$ r_4: $\langle$ *zhiyi* $\Rightarrow$ one of $\rangle$
r_5: $\langle$ *shaoshu guojia* X $\Rightarrow$ X the few countries $\rangle$

Figure 1: Synchronous hypergraph and source hypergraph of a sentence pair from Chinese segment "*shaoshu guojia zhiyi*" aligned with an English string "one of the few countries". These two hypergraphs are constructed in order to calculate the expectation in Eq. (5). Here, a hyperedge corresponds to an SCFG rule. A node is a bispan in synchronous hypergraph and is a source span in source hypergraph. Notably, the dotted derivation in Figure (b), which contains dotted hyperedges and goes through the source-span [1,3], is a "wrong" derivation, because it results in a translation inconsistent with the target string. Intuitively, we achieve the unsupervised learning by moving the probability of such "wrong" derivations in source hypergraph into the derivations in synchronous hypergraph.

Eq. (5) clearly shows that there are two expectations need to be calculated. The first one is the expectation $E_{p(\mathbf{d}|\mathbf{t},\mathbf{s})}[H_i]$ of a parameter given a sentence pair, and the second one $E_{p(\mathbf{d},\mathbf{t}|\mathbf{s})}[H_i]$ is the expectation of a parameter given the source sentence.

In the following sections, we first introduce how to use hypergraph to compute these expectations by synchronous hypergraph and source hypergraph respectively (Sec. 3.1). Then, we discuss the intractability of the exact training, and achieve tractable training by approximation (Sec. 3.2). Finally, we describe the training algorithm in detail to explain how the rules is induced (Sec. 3.3).

3.1 Inference with Hypergraph

We use hypergraph (Klein and Manning, 2001) to compactly represent the space of derivations. Based on hypergraphs, it's straightforward to calculate the two expectations in Eq. (5). The first expectation $E_{p(\mathbf{d}|\mathbf{t},\mathbf{s})}[H_i]$ is the expected value when observing both source sentence $\mathbf{s}$ and target sentence $\mathbf{t}$. The second expectation $E_{p(\mathbf{d},\mathbf{t}|\mathbf{s})}[H_i]$ is a similar function, but only the source sentence is observed. Thus, in order to calculate the first expectation, we construct a synchronous hypergraph to represent all derivations of a sentence pair. Similarly, for the second expectation, we use a source hypergraph to represent all derivations of a source sentence.

Figure 1 shows a synchronous hypergraph (a) and a source hypergraph (b). Each hyperedge is associated with an SCFG rule. In a synchronous hypergraph, a node is denoted by a nonterminal with a bispan. In contrast, a node in the source hypergraph is a nonterminal that spans

a continuous sequence of words of source sentence.

More formally, a **hypergraph** is a pair $\langle V, E \rangle$, where V is the set of **nodes**, E is the set of **hyperedges**. Each hyperedge $e \in E$ connects a set of antecedent nodes to a single consequent node. And each hyperedge corresponds to an SCFG rule r. In a **synchronous hypergraph**, a node $v \in V$ is in the form $X_{i,j,k,l}$, which denotes the nonterminal X spanning from i to j (that is $s_{i+1}...s_j$) in the source sentence, and from k to l in the target sentence. In a **source hypergraph**, each node $v \in V$ is in the form $X_{i,j}$, which spans from i to j in the source sentence.

Based on these hypergraphs, we compute the two expectations by applying the inside-outside algorithm as described in Li et al. (2009). The computation complexity is linear to the size of hypergraph $O(|E|)$. More exactly, $O(|E|)$ denotes $O(|\mathbf{s}|^3|\mathbf{t}|^3)$ for synchronous hypergraph, and $O(|\mathbf{G}||\mathbf{s}|^3)$ for source hypergraph. Here, $\mathbf{G}$ denotes all potential synchronous grammars.

3.2 Tractable Estimation by Approximation

However, the size of potential SCFGs $\mathbf{G}$ is extremely large given a vocabulary Ω, resulting in a large number of hyperedges in source hypergraph. See the rule r_2 in Figure 1. In reality, there are many potential translation rules that share the same source side "*shaoshu X*" as r_2, but with different target side. Suppose n is the maximum number of terminals in the target side, the number of such rules is up to $O(|\Omega|^n)$. All these rules can connect the source word "*shaoshu*" and the node $X_{1,2}$ to the node $X_{0,2}$, which produces a large number of incoming hyperedges for node $X_{0,2}$. Therefore, due to the large number of potential SCFGs, the number of hyperedges in a source hypergraph is very large. The computation on such a huge source hypergraph makes the exact inference intractable.

To make inference tractable and efficient, we shrink the size of source hypergraph by defining a smaller neighborhood with the synchronous hypergraph. We parse the source hypergraph using the neighbor grammar $\mathbf{NG}$ of synchronous hypergraph $\mathbf{H}$:

$$\mathbf{NG} = \text{Neighbor}(\mathbf{G}', \mathbf{H}) = \{r | r \in \mathbf{G}' \wedge src(r) \in \{src(r') | r' \in G(\mathbf{H})\}\} \quad (6)$$

Here, $\mathbf{G}'$ is the set of translation rules discovered in the training corpus by our algorithm. $src()$ denotes the source side of a rule. $G()$ is the set of rules in a hypergraph. The neighbor grammar contains those rules that are discovered during training (rather than all potential rules), and whose source sides occur in the synchronous hypergraph. Sine the size of $\mathbf{NG}$ is typical fairly small, the parsing of our source hypergraph becomes tractable in practice.

The definition of neighbor source hypergraph is inspired by contrastive estimation (Smith and Eisner, 2005). Similar shrinkage of discriminative neighborhood is also used in Dyer et al. (2011a). Notably, our approximation is consistent with the purpose of synchronous grammar induction for SMT. In SMT, the goal of grammar induction is for translation rather than synchronous parsing. Our source hypergraph corresponds to the potential translation space during SMT decoding. Thus, we expect such approximation to be suitable for SMT.

3.3 Training Algorithm

Based on the synchronous hypergraph and source hypergraph introduced above, we optimize $\mathbb{L}$ in an online style as shown in Algorithm 1. For each sentence pair, we first use cube-pruning based biparsing (Sec. 4) to construct a synchronous hypergraph $\mathbf{H}_1$ (line 4). Rules are discovered in the construction of hypergraph in $\mathbf{H}_1$. We then collect these learnt rules $G(\mathbf{H}_1)$ in $\mathbf{H}_1$,

Algorithm 1: Training($\{(\mathbf{s}^n, \mathbf{t}^n)\}_{n=1}^N$)

1 $\mathbf{G}' \leftarrow \varnothing$, $\lambda \leftarrow \mathbf{0}$
2 **for** $t = 0$ **to** T **do**
3 **for** $n = 0$ *to* N **do**
4 $\mathbf{H}_1 \leftarrow \textsc{BiParse}(\mathbf{s}, \mathbf{t})$ ▷ generate synchronous hypergraph
5 $\mathbf{G}' \leftarrow \mathbf{G}' + G(\mathbf{H}_1)$ ▷ collect grammars in $\mathbf{H}_1$
6 $\mathbf{NG} \leftarrow \textsc{Neighbor}(\mathbf{G}', \mathbf{H}_1)$
7 $\mathbf{H}_2 \leftarrow \textsc{ExhaustiveParse}(\mathbf{NG}, \mathbf{s})$ ▷ generate neighbor source hypergraph
8 $\lambda \leftarrow \lambda + \eta \times \frac{\partial \mathbb{L}}{\partial \lambda}(\mathbf{H}_1, \mathbf{H}_2)$ ▷ η is learning rate
9 **return** $\mathbf{G}'$, λ

and store them in $\mathbf{G}'$. After that, we create the neighbor source hypergraph $\mathbf{H}_2$ by exhaustive bottom-up chart parsing using the neighbor grammar **NG** (line 6). Finally, we calculate the gradient by these two hypergraphs and update the feature weights (line 8). When the algorithm is complete, we learn a grammar $\mathbf{G}'$ and also the feature weights λ of the model.

We implement an stochastic gradient descent (SGD) recommended by Bottou.[1] We schedule the learning rate η by an exponential decay (Tsuruoka et al., 2009). We set the regularization strength, initial learning rate and the base of exponential decay as 1.0, 0.2, 0.9 respectively. We choose these values by maximizing the translation performance measured by BLEU on the NIST 2002 development set with a subset of our training data including 20k sentence pairs.

4 Cube Pruning-based Synchronous Parsing

The approximation makes the training algorithm tractable. However, there is still one problem: how to efficiently construct the synchronous hypergraph? Exhaustive synchronous parsing requires $O(|\mathbf{s}|^3|\mathbf{t}|^3)$ time.[2] To overcome this challenge, we propose a novel and efficient cube-pruning based synchronous parsing algorithm. Given a sentence pair, this algorithm constructs the synchronous hypergraph and discovers rules in the sentence pair.

Instead of enumerating all possible hyperedges, we only create k-best hyperedges for each source span. This is achieved by exploiting the substructure in a hyperedge. For example, the hyperedge in Figure 2(b) contains two substructures: alignment of source sub-span $X_{0,2}$ and alignment of the third source word s_3. We maintain sorted candidate lists for every substructure, and create cubes that represent the potential combinations of these candidates (see Fig. 2(a)). In this way, we are able to create k-best hyperedges by cube pruning.

More specifically, we maintain charts for source words and source spans respectively:

- **s-chart** for each source word. The cell $chart[s, i]$ in s-chart stores a list of candidate alignments for the i-th source word. A candidate alignment is represented by a list of target index. The vertical dimension in Figure 2(a) is an instance of $chart[s, 3]$.

[1] http://leon.bottou.org/projects/sgd

[2] Dyer (2010) has shown that two monolingual parses can be more efficient than one synchronous parse, due to the sparsity of pre-fixed translation rules. Such rules are extracted by the heuristic two-step pipeline. In contrast, there are no pre-fixed translation rules in our case.

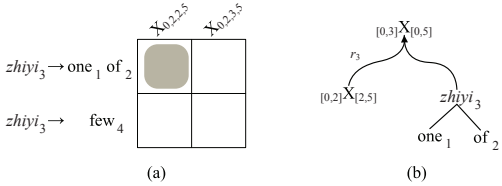


Figure 2: Construct hyperedge from a cube. (a) Cube $X_{0,2}$*zhiyi*$_3$ for source span (0,3). The vertical direction represents the candidate list of *zhiyi*$_3$, while the horizontal direction is a list of nodes that share the same source span (0,2). (b) the hyperedge corresponds to the gray grid in cube (a).

- **X-chart** for each source span. The cell $chart[X,i,j]$ in X-chart stores a list of nodes of the synchronous hypergraph. The nodes in $chart[X,i,j]$ share the same source span (i,j). The horizontal direction in Figure 2(a) is an instance of $chart[X,0,2]$.

These two charts are created by cube-pruning from the bottom up. When processing a source span, we infer all potential source parses and create cubes as shown in Figure 2(a) for every source parses. A cube represents the space of hyperedges for a source parse, where each dimension of a cube denotes the candidate list of a substructure for a hyperedge. Thus, the point in a cube corresponds to a hyperedge (see Fig. 2(b)). Based on these cubes, we create k-best hyperedges, store them into the chart, and then proceed to larger source spans.

4.1 Cube

Formally, a **cube** is a tuple of lists $\mathbf{L} = \langle L_1, ..., L_d, ..., L_{|L|} \rangle$, where $|L|$ is the dimension of the cube and L_d is a list for the d-th dimension. Although $|L|$ can be larger than 3 (a hypercube is defined in that case), we still call it cube for consistency. Given a point $\mathbf{p}$ in the cube, multiplication operator $\otimes$ constructs a hypothesis by $L_1[p_1] \otimes ... \otimes L_{|L|}[p_{|L|}]$. Particularly, there are two types of cubes: *word cube* and *span cube*.

Word Cube A word cube represents the alignment space for a source word s_i, and is used to create the items for $chart[s,i]$. A word cube has $|\mathbf{t}|$ dimensions. The d-th dimension list L_d contains two elements: ε (null translation) and t_d (the d-th target word). The hypothesis of a point $\mathbf{p}$ is a set of target words that aligns to the source word.

Span Cube A span cube represents the space of hyperedges for a source parse of source span (i,j), and is used for creating $chart[X,i,j]$. As a span cube denotes a deduction of the source span, it is represented by a source sequence of symbol γ, which is a mixture of words and nonterminals. For example, the cube in Figure 2(a) is represented by $X_{0,2}$*zhiyi*$_3$. According to γ, the d-th dimension L_d of a span cube $\mathbf{L}$ is defined according to γ_d:

$$L_d = \begin{cases} chart[s,i_1] & \text{if} \quad \gamma_d = s_{i_1} \\ chart[X,i_1,j_1] & \text{if} \quad \gamma_d = X_{i_1,j_1} \end{cases} \tag{7}$$

The hypothesis of a point $\mathbf{p}$ in a span cube corresponds to a hyperedge. The tail nodes of such a hyperedge is the set of nodes $\{v | v \in \{L_d[p_d]\}\}$, where $\{L_d[p_d]\}$ is the set of elements for

Algorithm 2: BiParse($\mathbf{s}, \mathbf{t}$)

```
1  for i ← 1,.., |s| do
2      for j ← 1,.., |t| do                          ▷ create a |t|-dimension word cube
3          L_j ← {ε, t_j}
4      𝒮 ← ⟨L_1, ..., L_|t|⟩
5      chart[s, i] ← MergeProducts(𝒮, ⊗)              ▷ create k-best alignments
6  for h ← 1,.., |s| do                              ▷ h is the size of span
7      forall the i, j s.t. j − i = h do
8          𝒮 ← ∅
9          for γ inferable from chart do             ▷ create all inferable span cubes
10             𝒮 ← 𝒮 + ⟨chart[γ_1], ..., chart[γ_|γ|]⟩
11         chart[X, i, j] ← MergeProducts(𝒮, ⊗)       ▷ create k-best hyperedges
```

the point $\mathbf{p}$ in the cube. The head node of the hyperedge $X_{i,j,k,l}$ has a target span that is the maximum target span covered by $\{L_d[p_d]\}$. See the Figure 2(b), for point (0,0), since the word "*zhiyi*$_3$" aligns to "one$_1$" and "of$_2$", and the tail node is $X_{0,2,2,5}$, the max target coverage is $(0,5)$. Therefore the head node is $X_{0,3,0,5}$.

However, for a point $\mathbf{p}$ in source cube, the elements in each dimension $\{L_d[p_d]\}$ may conflict with each other. See Figure 2(a). Point (0,1) in the cube denotes two elements: an alignment for "*zhiyi*$_3$" to "*few*$_4$" and a node $X_{0,2,2,5}$. The target alignment of "*zhiyi*$_3$" overlaps with the target span of node $X_{0,2,2,5}$. We say this is a *conflict*. Such a conflict denotes an inconsistent alignment. Therefore, we skip those conflicting points in a cube when constructing hyperedge.

Notably, a point in a span cube clearly expresses the inside word alignments of a hyperedge, allowing us to encode word level features for hyperedges.

4.2 Biparsing Algorithm

Algorithm 2 shows the main process of our cube-pruning based biparsing algorithm.

The algorithm begins with the construction of s-chart (lines 1-5). We first construct a $|\mathbf{t}|$-dimension word cube for every source word (lines 2-3). Secondly, by MergeProducts function as described in Chiang (2007), we create k-best alignments for every source word, and add them to the $chart[s,i]$ (line 5).

After that, we create each cell $chart[X,i,j]$ for X-chart (lines 6-11) from the bottom up. For a source span (i,j), all span cubes inferable from s-chart and X-chart are established. Every span cube is a tuple of $|\gamma|$-lists according to Eq. (7), and is append to $\mathscr{L}$ (lines 9,10). After creating all cubes, we construct k-best hyperedges and store them into $chart[X,i,j]$ (line 11).

Obviously, hyperedges and nodes have been created during the construction of the two charts, which means Algorithm 2 produces a synchronous hypergraph. As each hyperedge corresponds to an SCFG rule, Algorithm 2 is also a procedure to discover SCFG rules. On the other hand, the complexity of our biparsing algorithm is $O(|\mathbf{s}||\mathbf{t}| + |\mathbf{s}|k\log k + |\mathbf{s}|^3 + |\mathbf{s}|^2 k\log k)$. Considering k is prefixed and $|\mathbf{t}|$ is linear to $|\mathbf{s}|$ in reality, the complexity can be simplified into $O(|\mathbf{s}|^3)$.

4.3 Scoring Function and Future Cost

The hypothesis, that is created by the operator $\otimes$, corresponds to a hyperedge e and an SCFG rule r. Suppose the head node of a hyperedge e is $X_{i,j,k,l}$, we calculate the score of e by:

$$S(e) = \sum_i \lambda_i h_i(r, \mathbf{t}, \mathbf{s}) + F(i,j,k,l) \tag{8}$$

The first part is the feature score, while the second part $F()$ denotes the future cost. This future cost is an approximation of the outside score of bispan $[i,j,k,l]$, which denotes the cost to generate strings outside the head node $X[i,j,k,l]$. We estimate the future cost by using word level features as follows:

$$F(i,j,k,l) = \sum_{g \notin (i,j)} \operatorname*{argmax}_{h \notin (k,l)} S(s_g, t_h) + \sum_{h \notin (k,l)} \operatorname*{argmax}_{g \notin (i,j)} S(s_g, t_h) \tag{9}$$

$S(s_g, t_h)$ calculates the score of features for a word pair. The future cost can be calculated before inference.

Notably, a rule is not factorized into hypothesis in each dimension of a cube. Because rule features are used in our model, the multiplication operator $\otimes$ is only approximately monotonic under this scoring function. Thus, the MERGEPRODUCTS only produces approximate k-best hypotheses.

4.4 Implementation

Here, we discuss some engineering details of our algorithm.

For efficiency, we only construct 30-best hyperedges for each cube. These hyperedges are recombined, and the top-10 nodes are saved to the chart.

We prune the SCFG rules (corresponding to hyperedge) that are already discovered from synchronous hypergraphs using two constraints. Firstly, we prune a hypergraph by viterbi pruning with $p = 1$ (Huang, 2008). Secondly, we maintain a rule cache with size limit of 20M. When the cache is full, we drop those rules containing nonterminals that are discovered in only one sentence pair. For rules sharing the same source side, we retain the 50 most frequent ones.

Sometimes, a hyperedge produced by the cube corresponds to an illegal SCFG rule as defined in Chiang (2007). For example, rules without source terminals, target terminals and ITG rules (Dekai, 1997). We allow such edges to be added to the hypergraph, in order to make sure that most sentences are reachable. By this setting, the failure of biparsing only occurs in less than 1% of the whole training corpus. However, due to the ambiguity of these rules, we do not use them during end-to-end translation.

Overall, by these implementations, our synchronous algorithm (Alg. 2) runs in a speed of 0.41 second/sentece-pair.

5 Features

The log-linear model allows us to incorporate arbitrary features. In addition to the rules that we are interested in, we also incorporate word-level information and translation boundary information, both of which have proved useful for SMT. Through these features, we can enhance the ability of our model to predict whether a translation derivation is good or not. Notably, previous work typically use such information in locally normalization, while we learn weights of these features by globally normalization on the entire sentence structure.

Rule features We associate each rule with a single feature. Each rule feature counts the number of times that a rule appears in a derivation. In this way, we are able to learn a weight for every rule by global normalization. This is quite different from traditional pipeline that gives each rule with relative frequency that is locally normalized by rules with the same source side or target side.

Word association features As shown in section 4, our inference algorithm also induces inside word alignments for each derivation. Thus, it is straightforward to introduce features that contain word level information. Here, we associate each word pair with a fine-grained boolean feature. We also include the two lexical weights as described by Koehn et al. (2003), estimated by word translation probabilities output by GIZA++. We set the initial weight of these two lexical weights as 1.0. The lexical weights enable our system to score and rank the hyperedges at the beginning. Although word alignment features are used, our system is neither constrained by prefixed word alignment nor requires heuristic alignment combination strategy.

Phrase Boundary features As shown in Figure 1(b), although the rule sequence of the solid derivation and dotted derivation are exactly the same, only solid derivation produces correct translation. They can be distinguished by taking into account boundary information (Xiong et al., 2010; Dyer et al., 2011b). We design two feature templates here: $BE{:}s_i + 1, s_j$ denotes the bigrams of beginning and ending source words. $PS{:}s_i, s_{j+1}$ means the preceding and succeeding source words. We do not use target word information, since this is non-local for hypergraph, and will largely increase the size of hypergraph.

Length Feature Finally, we also integrate the length feature of target side that is used in traditional SMT system.

6 Experiments

In this section, we present our experiments on Chinese-to-English translation tasks. The experiments are aimed at measuring the quality and effectiveness of grammar induced by our method. We present the performance of our unsupervised discriminate synchronous grammar induction (UDSGI) using two groups of features during decoding. We test the translation performance of UDSGI and the baseline on the same decoder.

Baseline The baseline system is an in-house implementation of hierarchical phrase-based translation system(Chiang, 2007). The grammar is extracted from word-aligned corpus by traditional two-step pipeline. Symmetric word alignments were created by first running GIZA++ (Och and Ney, 2003) in both directions and then applying refinement rule "grow-diag-final-and" (Koehn et al., 2003). The system uses 8 dense features including: forward and backward translation probabilities; forward and backward lexical weights; language model; 3 penalties for word count, extracted rule count, and glue rule count.

UDSGI Dense This configuration uses the same feature set as the baseline. Our log-linear model for grammar induction does not contain the forward and backward translation probabilities. However, we still compute the forward and backward translation probabilities for UDSGI by normalizing the expectation $E_{p(\mathbf{d}|\mathbf{t},\mathbf{s})}[H_i]$ of a rule, when algorithm 1 is complete. The normalization is similar to the traditional estimation of these two probabilities.

	|Source Words|		|Target Words|		Number of rules	
Rule Type	Baseline	UDSGI	Baseline	UDSGI	Baseline	UDSGI
Phrase	3.16	2.39	3.97	2.59	2.0M	6.6M
One-NT	3.20	2.31	3.91	2.51	6.4M	2.9M
Two-NT-Mono	2.62	2.52	3.47	2.81	4.2M	2.7M
Two-NT-Swap	2.55	2.46	3.65	2.87	0.5M	0.8M
All Rule	2.89	2.40	3.77	2.63	13.2M	13.1M

Table 1: Comparison of grammars induced by Baseline and UDSGI. Phrase represents the phrase rule. One-NT and two-NT denotes rules contain one or two nonterminals. Mono and Swap means that the two nonterminals are monotone or swapping in the target side. |Source Words| and |Target Words| denotes the average count of source and target terminals in a rule.

UDSGI Dense+Sparse Our global log-linear model encodes millions of sparse features including rule, word pair and phrase boundary features. During decoding, we also enhance our model by these sparse features. In order to optimize these sparse features with the dense features by minimum error rate training (MERT), we group features of the same type into one coarse "summary feature", and get three such features including: rule, word-pair and phrase-boundary features. In this way, we rescale the weights of the three "summary features" with the 8 dense features by MERT. Such approach is similar to Dyer et al. (2011b).

6.1 Data

We used the NIST evaluation set of 2002 (MT02) as our development set for MERT, and test the translation performance on the NIST 2003-2005 (MT03-05) evaluation sets. Case-insensitive NIST BLEU-4 (Papineni et al., 2002) is used to measure translation performance.

We used a bilingual corpus that contains 200K sentence pairs of up to length 40 from the LDC data. [3] There are 8.8M words in the 200K data. We trained a 5-grams language model by the SRILM toolkit (Stolcke, 2002). The monolingual data for language model training includes the Xinhua portion of the GIGAWORD corpus and the English side of the entire LDC data, which contains 432 million words.

We used MERT (Och, 2003) to optimize the feature weights for decoding by maximizing BLEU. Since the instability of MERT has a substantial impact on results, we follow Clark et al. (2011) to report the average scores of 3 independent runs.

6.2 Grammar Analysis and Oracle Results

The synchronous grammar generated by UDSGI has 13.1 millions rules. The number of these rules is comparable with that of grammar extracted by the traditional pipeline, which has 13.2 millions rules. However, the two grammars are quite different as shown in Table 1. Our grammar is more reusable than the baseline's, because it contains less source words and less target words. Interestingly, even we discard the rules which occur only once (See Sec. 4.4), we still learn 60% more swapping rules with two nonterminals (Two-NT-Swap) than the baseline.

[3]The 200K data comes from the following corpus: LDC2002E18, LDC2003E07, LDC2003E14, LDC2004T07, LDC2005T06 and Hansards portion of LDC2004T08 (part of).

System	MT03	MT04	MT05	Avg.
Baseline	64.96	69.48	65.45	66.63
UDSGI	73.37	77.13	74.29	74.93

Table 2: BLEU-4 scores of oracle translations generated by grammars of Baseline and UDSGI. *Avg.* reports the average score on the three test set.

System	$\|\mathbf{G}'\|$	MT03	MT04	MT05	Avg.
Moses-Chart	45.0M	32.93	34.73	31.24	32.97
Baseline	13.2M	32.36	34.51	31.86	32.91
Baseline-expand	46.2M	33.04	35.13	32.08	33.42
UDSGI Dense	13.1M	33.46	35.43	32.74	33.87
UDSGI Dense+Sparse	13.1M	33.58	36.27	33.05	34.30

Table 3: Evaluation of translation quality in terms of BLEU. *Moses-Chart* is the running of hierarchical phrased-model in Moses. *Baseline-expand* uses a similar extraction constraint to Moses-Chart and the same decoder as Baseline. The improvement of UDSGI over Baseline is statistically significant (Koehn, 2004) ($p < 0.01$).

We can not directly evaluate the quality of grammar, since there is not a golden grammar. However, as grammar is used to generate target translations, it's reasonable to decide the quality of a grammar by testing what best translations it can produce. Therefore, we compare the oracle translation result of the grammar in baseline and the grammar in UDSGI, with four reference translations given.[4] As shown in Table 2, our grammar achieves a much higher oracle BLEU (+8.3 BLEU points) than the baseline grammar. This suggests that the grammar induced by our method is able to generate better translations than the grammar extracted by the traditional two-step pipeline.

6.3 Translation Results

Table 3 compares the translation performance of our approach and the baseline on the test sets. When extracting rules as Chiang (2007), the baseline produces an average BLEU of 32.91. We also run Moses-Chart, the implementation of hierarchical phrased-model in Moses (Koehn et al., 2007), on our data with its default settings. As shown in the table, our Baseline is comparable with Moses-Chart. However, Moses-Chart extracts much more rules, because it uses a different extraction constraint from Chiang (2007). The differences mainly include edges of initial phrases can be unaligned and minimum size of source part of sub-phrases is 2.

We also relax the Baseline by applying these two options, and call such setting as *Baseline-expand*. Baseline-expand outperforms Baseline by +0.51 BLEU with 3.5 times of rules. As our UDSGI method learns a similar size of rules with the Baseline's, we only compare our method with the Baseline in the following sections.

Using the same 8 dense features as used in the baseline system, we obtain an average improve-

[4]We calculate the oracle translation by the traditional decoder using sentence BLEU score as feature function. We use add 0.1 smoothing for the ngram precision, and set the reference length for a source span (i,j) as $\lceil \frac{L_{min} \times (j-i)}{|\mathbf{s}|} \rceil$. Here, L_{min} is the shortest length of references for a sentence.

System	$\|G'\|$	MT03	MT04	MT05	Avg.
Baseline	17.6M	33.83	35.94	33.23	34.33
UDSGI Dense+Sparse	13.5M	35.28	37.43	34.25	35.65

Table 4: Experiment results by integrating the entire LDC corpus.

ment of +0.96 BLEU score on the three test sets over the Baseline. As the difference between Baseline and UDSGI Dense only lies in the grammar, the improvement indicates that the grammar induced by UDSGI does outperform that extracted by the traditional two-step pipeline. When we incorporate the sparse features learnt by our training algorithm, we achieve a further improvement of +0.43 BLEU point. Therefore, our training algorithm is able to learn the useful information encoded by the sparse features for translation.

6.4 Exploiting the Entire LDC Data

Since the previous experiment only runs on a medium-scale corpus, we want to know whether the improvement only comes from poor word alignment performance in the baseline system. Therefore, we want to verify the improvement on large-scale data set. Unfortunately, our approach needs to store all rules in memory, which is impractical for large scale data. Instead of directly training on LDC data (357M words), which is 40-times larger than the medium-scale data, we try to explore these data in a different way.

The baseline still extracts rules from the 200K data, while our approach also learns grammar on 200K data. However, we run GIZA++ on the entire LDC data. The word alignment performance of the baseline is therefore enhanced by using all entire data. We used the word translation probabilities of this large-scale trained GIZA++ to calculate lexical weights in our model during training.

Table 4 shows the results. Not surprisingly, the performance of baseline significantly increases by 1.4 points, due to the improvement of word alignment quality and the larger number of extracted rules. Interestingly, our method also achieves similar improvements with enhanced GIZA++. Still, our approach outperforms the baseline by +1.3 points, which is quite close to the improvement in medium-scale corpus experiments. Furthermore, the grammar in UDSGI is smaller than the grammar of the baseline (76.7%). Therefore, we believe that our improvement comes from the theoretical advantage of our approach, and that such advantage does exist even on large-scale data.

7 Related Work

Because of the great importance of synchronous grammars to SMT systems, researchers have proposed various methods in order to improve the quality of grammars. In addition to the generative model introduced in Section 1, researchers also have made efforts on word alignment and grammar rescoring.

The first effort is to improve word alignment by considering phrase/syntax information (May and Knight, 2007; DeNero and Klein, 2010; Pauls et al., 2010; Burkett et al., 2010; Riesa et al., 2011). Such approaches also use discriminative framework to combine word alignment and syntactic alignment information. In this way, they prefer word alignments that are consistent with syntactic structure alignments. However, labeled word alignment data are required in order to learn the discriminative model.

Researchers also try to rescore the weights of translation rules. They rescore the weights of rules extracted from the two-step pipeline, by using the similar latent log-linear model as us (Blunsom et al., 2008; Kääriäinen, 2009), or incorporating various features using labeled data (Huang and Xiang, 2010). Such methods still need to run the heuristic two-step pipeline to extract the grammar, while our method can directly learn the grammar and correspondent weights.

Our work also has a connection to the research direction that exploits resources-rich languages to construct similar tools for resource-poor languages. This can be done with parallel data (Pauls et al., 2010) or without parallel data (Cohen et al., 2011).

Saers et al. (2009) also propose a cubic biparsing algorithm based on beam pruning. They apply this algorithm for generative model-based ITG grammar induction.

Conclusion and Future Work

We have presented a global log-linear model for synchronous grammar induction, and have also proposed efficient training and inference algorithms. In addition to the theoretical advantage, we also achieve significant improvements over the traditional heuristic two-stage pipeline on both medium-scale and large-scale training data.

In the future, we hope to find efficient algorithms that are capable of incorporating contextual features, especially language model and context-based translation model, and optimizing parameters by maximizing BLEU. Because our proposed model is quite general, we are also interested in applying this method to linguistically syntax-based SMT.

Acknowledgments

The authors were supported by High-Technology R&D Program (863) Project No. 2011AA01A207 and No. 2012AA011102, and NSFC Project No. 60903138 and No. 61202216. We would like to thank Yi Lin and the anonymous reviewers for their insightful comments.

References

Blunsom, P., Cohn, T., Dyer, C., and Osborne, M. (2009). A gibbs sampler for phrasal synchronous grammar induction. In *In Proc. ACL 2009*.

Blunsom, P., Cohn, T., and Osborne, M. (2008). A discriminative latent variable model for statistical machine translation. In *Proc of ACL-08*.

Burkett, D., Blitzer, J., and Klein, D. (2010). Joint parsing and alignment with weakly synchronized grammars. In *Proc. NAACL 2010*.

Cherry, C. and Lin, D. (2007). Inversion transduction grammar for joint phrasal translation modeling. In *Proc. SSST 2007, NAACL-HLT Workshop on Syntax and Structure in Statistical Translation*.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proc. 2011*, pages 176–181.

Cohen, S. B., Das, D., and Smith, N. A. (2011). Unsupervised structure prediction with non-parallel multilingual guidance. In *Proc. EMNLP 2011*.

Cohn, T. and Blunsom, P. (2009). A Bayesian model of syntax-directed tree to string grammar induction. In *Proc. EMNLP 2009*.

Dekai, W. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

DeNero, J., Bouchard-Côté, A., and Klein, D. (2008). Sampling alignment structure under a Bayesian translation model. In *Proc. EMNLP 2008*.

DeNero, J. and Klein, D. (2010). Discriminative modeling of extraction sets for machine translation. In *Proc. ACL210*.

Dyer, C. (2010). Two monolingual parses are better than one (synchronous parse). In *Proc. NAACL 2011*.

Dyer, C., Clark, J. H., Lavie, A., and Smith, N. A. (2011a). Unsupervised word alignment with arbitrary features. In *Proc. ACL2011*.

Dyer, C., Gimpel, K., Clark, J. H., and Smith, N. A. (2011b). The cmu-ark german-english translation system. In *Proc. WMT2011*.

Galley, M., Graehl, J., Knight, K., Marcu, D., DeNeefe, S., Wang, W., and Thayer, I. (2006). Scalable inference and training of context-rich syntactic translation models. In *Proc. ACL 2006*.

Huang, F. and Xiang, B. (2010). Feature-rich discriminative phrase rescoring for smt. In *Proc. Coling 2010*.

Huang, L. (2008). Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08*.

Kääriäinen, M. (2009). Sinuhe – statistical machine translation using a globally trained conditional exponential family translation model. In *Proc. EMNLP 2009*.

Klein, D. and Manning, C. D. (2001). Parsing and hypergraphs. In *Proc. IWPT 2001*.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *Proc. EMNLP 2004*.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proc. ACL 2007 (demonstration session)*.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proc. HLT-NAACL 2003*.

Levenberg, A., Dyer, C., and Blunsom, P. (2012). A bayesian model for learning scfgs with discontiguous rules. In *Proc. EMNLP 2012*. Association for Computational Linguistics.

Li, Z., Eisner, J., and Khudanpur, S. (2009). Variational decoding for statistical machine translation. In *Proc. ACL 2009*.

Liu, Y., Liu, Q., and Lin, S. (2006). Tree-to-string alignment template for statistical machine translation. In *Proc. ACL 2006*.

Marcu, D. and Wong, W. (2002). A phrase-based, joint probability model for statistical machine translation. In *Proc. EMNLP 2002*.

May, J. and Knight, K. (2007). Syntactic re-alignment models for machine translation. In *Proc. EMNLP 2007*.

Neubig, G., Watanabe, T., Sumita, E., Mori, S., and Kawahara, T. (2011). An unsupervised model for joint phrase alignment and extraction. In *Proc. ACL 2011*.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proc. ACL 2003*.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL 2002*.

Pauls, A., Klein, D., Chiang, D., and Knight, K. (2010). Unsupervised syntactic alignment with inversion transduction grammars. In *Proc. NAACL 2010*.

Riesa, J., Irvine, A., and Marcu, D. (2011). Feature-rich language-independent syntax-based alignment for statistical machine translation. In *Proc. EMNLP 2011*.

Saers, M., Nivre, J., and Wu, D. (2009). Learning stochastic bracketing inversion transduction grammars with a cubic time biparsing algorithm. In *Proc. IWPT 2009*.

Smith, N. A. and Eisner, J. (2005). Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. ACL 2005*.

Stolcke, A. (2002). Srilm – an extensible language modeling toolkit.

Tsuruoka, Y., Tsujii, J., and Ananiadou, S. (2009). Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proc. ACL 2009*.

Xiao, X., Liu, Y., Liu, Q., and Lin, S. (2011). Fast generation of translation forest for large-scale smt discriminative training. In *Proc. EMNLP 2011*. Association for Computational Linguistics.

Xiong, D., Zhang, M., and Li, H. (2010). Learning translation boundaries for phrase-based decoding. In *Proc. NAACL2010*.

Zhang, H., Quirk, C., Moore, R. C., and Gildea, D. (2008). Bayesian learning of non-compositional phrases with synchronous parsing. In *Proc. ACL 2008*.