Better Learning and Decoding for Syntax Based SMT Using PSDIG

Yuan Ding **Martha Palmer**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
{yding, mpalmer}@cis.upenn.edu

Abstract

As an approach to syntax based statistical machine translation (SMT), Probabilistic Synchronous Dependency Insertion Grammars (PSDIG), introduced in (Ding and Palmer, 2005), are a version of synchronous grammars defined on dependency trees. In this paper we discuss better learning and decoding algorithms for a PSDIG MT system. We introduce two new grammar learners: (1) an exhaustive learner combining different heuristics, (2) an n-gram based grammar learner. Combining the grammar rules learned from the two learners improved the performance. We introduce a better decoding algorithm which incorporates a tri-gram language model. According to the Bleu metric, the PSDIG MT system performance is significantly better than IBM Model 4, while on par with the state-of-the-art phrase based system Pharaoh (Koehn, 2004). The improved integration of syntax on both source and target languages opens door to more sophisticated SMT processes.

1 Introduction

Syntax-based statistical machine translation (SMT) aims at applying statistical models to structural data, has begun to emerge. With the advances in the broad-coverage parsers trained from treebanks, e.g. (Collins, 1999) and (McDonald et al., 2005), structural analysis of different languages has been made possible. Ideally, by combining the natural language syntax and machine learning methods, a broad-coverage and linguistically well-motivated statistical MT system can be constructed.

However, because of structural divergences between languages (Dorr, 1994)，due to either systematic differences between languages or loose translations in real corpora, the syntax based MT systems have to transduce between non-isomorphic tree structures, which is a major challenge.

(Wu, 1997) and (Alshawi et al., 2000) learn the tree representations directly from parallel sentences, and do not make allowances for non-isomorphic structures. (Yamada and Knight, 2001) used a sequence of tree operations transforming a syntactic tree into a string of the target language. (Quirck et al., 2005) used syntax structures to guide the phrase reorder process.

When researchers try to use syntax trees (here we think of syntax as linguistic syntax, in contrast to formal syntax) in both languages, the problem of non-isomorphism must be addressed. In theory, stochastic tree transducers and some versions of synchronous grammars provide solutions for the non-isomorphic tree based transduction problem and hence possible solutions for MT. Synchronous Tree Adjoining Grammars was proposed by (Shieber and Schabes, 1990). Eisner (2003) proposed viewing the MT as synchronous tree substitution grammar parsing. Melamed (2003) formalized the MT as synchronous parsing based on multi-text grammars. Graehl and Knight (2004) defined training and decoding algorithms for both generalized tree-to-tree and tree-to-string transducers. Lin (2004) proposed to base the MT process on parallel dependency paths. Ding and Palmer (2005) introduced the use of Probabilistic Synchronous Dependency Insertion Grammars (PSDIG) to model machine translation. These approaches model the two languages using tree transduction rules or synchronous grammars, possibly with multi-lemma elementary structures as atomic units.

However, large scale implementation and competitive performance of the above mentioned methods are still a challenging task. And to the best of our knowledge, the advantages of syntax based statistical MT systems over pure statistical MT systems are yet to be empirically verified.

We believe difficulties in inducing a synchronous grammar or a set of tree transduction rules from large scale parallel corpora are caused by:

1. A synchronous derivation process must exist between the source and target language sentences. However identifying the proper level for the transduction is not an easy task.
2. The induction of a synchronous grammar is usually computationally expensive. The exhaustive search for all possible corresponding sub-sentential structures is NP-complete.
3. The problem is aggravated by the non-perfect training corpora. Loose translations are less of a problem for string based approaches than for approaches that require syntactic analysis.

Hajic et al. (2002) limited non-isomorphism by n-to-m matching of nodes in the two trees. However, even by allowing cloning operations on subtrees, Gildea (2003) found that parallel trees over-constrained the alignment problem, and achieved better results with a tree-to-string model. In a different approach, Hwa et al. (2002) aligned the parallel sentences using SMT models and projected the alignments back to the parse trees.

The framework of PSDIG, while using trees on both languages, achieves flexible transduction of non-isomorphic trees by (1) relying on dependency structure, which is a deeper form of representation compared to phrasal structure trees; and (2) by allowing multi-lemma elementary trees. The linear time decoding algorithm in (Ding and Palmer, 2005) can be used as a starting point for more sophisticated decoding. In this paper, we examine various aspects of syntax based MT using PSDIG, such as grammar induction and decoding algorithms. We aim at having the overall MT system perform competitively with the pure statistical MT systems, especially the current leading SMT system based on phrasal translation, Pharaoh (Koehn, 2004).

The rest of this paper describes the system details as follows: Sections 2 sketches the concept of PSDIG. Section 3 describes two new algorithms to induce Synchronous Dependency Insertion Grammars from parallel corpora. Section 4 describes a new decoder which incorporates several translation models. We evaluate our system in section 5 with the Bleu metric (Papineni et al., 2002) and discuss the results in Section 6.

2 A Sketch of PSDIG

The framework of using PSDIG for MT was introduced in (Ding and Palmer, 2005). Fox (2002) reports dependency representations have the best inter-lingual phrasal cohesion properties. Furthermore, dependency grammars have the advantage of simple formalism and CFG equivalent formal generative capacity. Being lexicalized, dependency grammars are friendly with probabilistic modeling.

A monolingual Dependency Insertion Grammar (DIG) can be viewed as a tree substitution grammar defined on dependency trees (as opposed to phrasal structure trees). The basic units of the grammar are elementary trees (ET), which are sub-sentential dependency structures containing one or more lexical items. The synchronous version, Synchronous Dependency Insertion Grammar (SDIG), assumes that the isomorphism of the two syntactic structures is at the ET level, rather than at the word level, hence allowing non-isomorphic tree to tree mapping.

We illustrate how the SDIG works using the following pseudo-translation example (the placement of the dependency arcs reflects word order):

- [*Source*] *The girl kissed her kitty cat.*
- [*Target*] *The girl gave a kiss to her cat.*

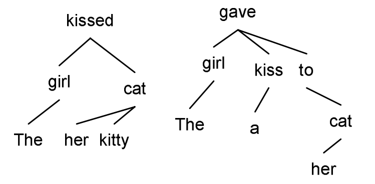


Figure 1. An example

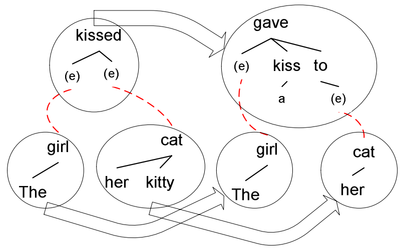


Figure 2. Tree-to-tree transduction

Almost any tree-transduction operations defined on a single node will fail to generate the target sentence from the source sentence without using insertion/deletion operations. However, if we view each dependency tree as an assembly of indivisible sub-sentential elementary trees (ETs), we can find a proper way to transduce the input tree to the output tree. An ET is a single "symbol" in a transducer's language. As shown in Figure 2, each circle stands for an ET and thick arrows denote the transduction of each ET as a single symbol.

3 Improving Synchronous Dependency Insertion Grammar Induction

As the start to a syntax-based SMT system, the PSDIG must be learned from the parallel corpora.

3.1 Background

One way to induce a generative grammar is using EM style estimation on the generative process (Hajic et al., 2002; Eisner 2003; Gildea 2003; Graehl and Knight 2004). However, Fox (2002) reported that between French and English, the percentage of head crossings dependencies per chance is 12.62%. In (Ding and Palmer, 2005), using the statistics from a small word to word aligned Chinese-English parallel corpus[1], it is found that crossing-dependencies between Chinese and English is 4.7% while broken dependencies (i.e. descendents go to sister nodes in another language) is 59.3%.

The large number of broken dependencies presents a major challenge for grammar induction based on a top-down style EM learning process.

In (Ding and Palmer, 2005), a grammar induction algorithm that hierarchically partitions the parallel dependency trees was introduced. The algorithm runs in an iterative fashion, in each step, it chooses more probable word mappings from the two dependency trees of the two languages, subject to certain category constraints and synchronously partitions the two dependency trees on the two sides at the chosen word pair.

The "synchronous partitioning" operation can be viewed as the reversed procedural of Figure 2. Suppose we synchronously partition at *word pairs* "*girl – girl*" and "*cat – cat*" the resultant three treelet pairs can be collected.

The probability of each word mapping is estimated using a max entropy model, based on a set of heuristic functions and trained on a development test dataset. The algorithm then runs as follows: (1) it looks at all the top-NP (noun phrases) mappings, and synchronously partition at more confident mappings; (2) it looks at normal NP pairs and partition the tree pair; (3) partition at all VPs (verb phrases) pairs and equivalents (4) partition at all the Modifiers (ADJP, PP, etc…); (5) all the numbers.

The above algorithm is greedy. It cannot correct previous errors. Suppose a wrong partitioning operation is taken at Step (1), all the partitioning operations taken in step (2) to (5) are subject to this error, and hence hurt the accuracy of the statistics.

Moreover, the above algorithm constraints the partitioning operations can only be taken between two nodes of the same category set. However, if a NP is mapped to a VP, which does happen in real world data, such constraints wouldn't allow it.

3.2 Grammar Induction by Exhaustive Search

In light of the previously observed problems of the hierarchical partitioning algorithm, we remove the category constraints of the grammar induction process. Rather, we only rely on the heuristic score provided by the Max Entropy model:

$$\begin{aligned}&\mathrm{P}(y \mid e_{i,} f_j)\\ &= \mathrm{P}\left(y \mid h_0(e_i, f_j), h_1(e_i, f_j) \ldots h_n(e_i, f_j)\right) \qquad (1)\\ &= \frac{1}{Z}\exp\left(\sum_k \lambda_k h_k(e_i, f_j) + \lambda_s\right)\end{aligned}$$

where $y = (0,1)$ as labeled in the training data whether the two words are mapped with each other. h_k are the heuristic functions (details given later).

We calculate the above probability for all the tentative word pairs; and filters them with a threshold, $\theta_{threshold}$, this means the word pairs that have a heuristic score higher than $\theta_{threshold}$ will be used in the exhaustive search for PSDIG induction. The value of $\theta_{threshold}$ is chosen by optimizing the F-measure on the development test data.

- [*English*] *I have been in Canada since 1947.*
- [*Chinese*] *Wo 1947 nian yilai yizhi zhu zai jianada.*
- [*Glossary*] *I 1947 year since always live in Canada*

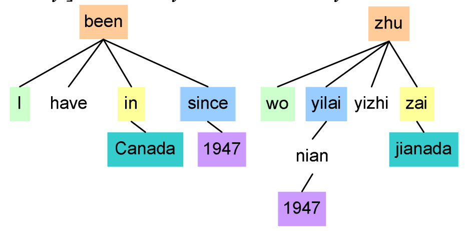


Figure 3a. A grammar induction example

[1] Total 826 sentence pairs, 9957 Chinese words, 12660 English words. Data made available by the courtesy of Microsoft Research, Asia and IBM T.J. Watson Research.

Suppose in the above example, the MaxEnt model and the threshold predict six *word pairs* should be used to collect the *treelet pairs*:

(I – wo), (Canada – jianada), (been – zhu), (in – zai), (since – yilai), (1947 – 1947).

For each of the permitted *word pair*, it has the freedom of going "on" and "off", suggesting a partitioning operation to be "taken" or "not taken" at this word pair. For each *combination* of the abovementioned on/off choices of the *word pairs*, a *synchronous partitioning* operation is executed at each *word pair* that is currently set as "on".

For each abovementioned *combination*, the resultant *treelet pairs* from the *synchronous partitioning* operations taken are collected as ET pairs. If we use four word pairs as "on" (out of six):

(I – wo), (been – zhu), (in – zai), (since – yilai)

The following four ET pairs will be collected:

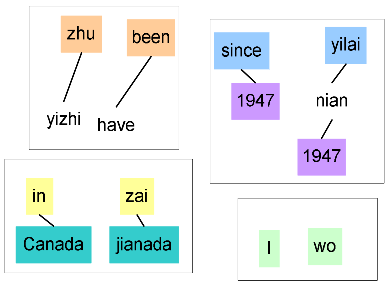


Figure 3b. Collected ETs

In the above example, the ET pairs corresponding to the string pairs below are learned as PSDIG rules:

(I have been in Canada since 1947 -- wo 1947 nian yilai yizhi zhu zai jianada) (have been -- yizhi zhu) (I have been -- wo yizhi zhu) (since 1947 -- 1947 nian yilai) (in -- zai) (Canada -- jianada) (have been since -- yizhi zhu zai) ……

In theory, by permitting n word pairs, 2^n *combinations* are possible. Hence an exponentially large number of ET pairs may be learned. We prune the ET pairs using the following:

- Any ET has a max size of $size_{\max}$, (currently $size_{\max} = 7$)
- The ETs on two sides have a max size ratio of $ratio_{size} : 1$, (currently $ratio_{size} = 4$)
- For all ET pairs that rooted at the same word pair, we only allow $c_{\max}$ distinctive ET pairs. We do so by allowing only less confident (lower heuristic score) word pairs to have the freedom of going "on" and "off", while we set the more confident word pair to be always "on", meaning a partitioning operation is always "taken" at more confident locations. (currently $c_{\max} = 1024$)
- Each unique ET pair is only counted once for a given sentence pair.

This exhaustive search algorithm provides more flexibility compared to the previous hierarchical partitioning algorithm. We observe that the number of rules learned using exhaustive search is significantly larger than that of those learned using hierarchical partitioning.

3.3 Heuristics for the Exhaustive Learner

The heuristic function that is used in Equation (1) is based on a set of heuristics. While some of the heuristics are the same as those in (Ding and Palmer, 2005), we build two new heuristic functions, incorporating word alignment results. For a word pair (e_i, f_j) for the tentative partitioning operation, the heuristics functions are described as following:

- Inside-outside word alignment scores. (new)
- Inside-outside word penalties. (new)
- Entropy: the entropy of the word to word translation probability of the English word e_i. The lower the entropy, the more selective and hence more reliable the word alignment.
- Part-of-Speech mapping template: whether the POS tags of the two words are in the "highly likely to match" POS tag pairs.
- Word translation probability: $P(f_j | e_i)$
- Whether the word pair is in word alignment.

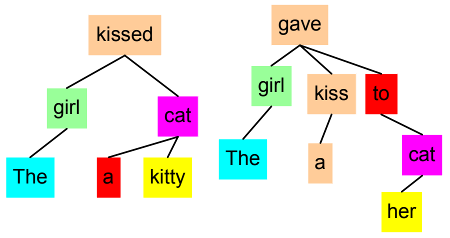
The first two heuristics are new in this paper. They are built using a word alignment table generated by bi-directional training of IBM Model 4 (Brown et al. 1993), using the method described in (Och and Ney, 2004). The alignments from models of both directions are intersected and diagonally grown and finalized (a.k.a. grow-diag-final).

Suppose we have the word alignment as shown in Figure 4.

Please be noted that since the align-grow-final method tends to align adjacent words diagonally, some alignments are not exact or not correct.

We define the subtree that is rooted at the word

e_i as the "inside tree" of e_i. And the rest of the dependency tree as the "outside tree" of e_i. This idea is borrowed from the inside-outside probabilities in PCFG parsing. For example, on the left side, the inside tree of (*cat*) is (*a kitty cat*), while the outside tree of (*cat*) is (*The girl kissed*).



(The – The), (girl – girl), (kissed – gave, a, kiss), (to – a), (cat – cat), (kitty – her).

Figure 4. A word alignment example

With regards to the tentative word pair (cat – cat), we calculate the number of word alignments from the inside tree on the left side to the inside tree on the right side. Hence, counting (cat – cat), (kitty – her), the inside word alignment score is:

$$h_{score\_inside}(\text{cat,cat})=2$$

Likewise we define the outside word alignment score being the number word alignments from the outside tree on the left side to the outside tree on the right side. Counting (The – The), (girl – girl), (kissed – gave, a, kiss), we have:

$$h_{score\_outside}(\text{cat,cat})=5$$

Also, word alignments that violate inside-outside tree consistency are counted as a penalty term, hence, counting (to – a), we have:

$$h_{penalty\_in-out}(\text{cat,cat})=1$$

Formally, given *alignment* as the word alignment between two sentences, T_e, T_f being any two treelets on each side, define:

$$\text{Count}(T_e, T_f) = \sum_{e_i \in T_e, f_j \in T_f, (e_i, f_j) \in alignment} 1 \qquad (2)$$

Let $InT(x)$, $OutT(x)$ be the inside tree and out side tree of word x, respectively. We have:

$$h_{score\_inside}(e_i, f_j) = \text{Count}\left(InT(e_i), InT(f_j)\right) \qquad (3)$$

$$h_{score\_outside}(e_i, f_j) = \text{Count}\left(OutT(e_i), OutT(f_j)\right) \qquad (4)$$

$$h_{penalty\_in-out}(e_i, f_j) = \text{Count}\left(OutT(e_i), InT(f_j)\right) + \text{Count}\left(InT(e_i), OutT(f_j)\right) \qquad (5)$$

The above heuristics are a set of real valued numbers. We use a Maximum Entropy model to log-linearly interpolate the heuristics, as in (1). The MaxEnt model is trained using the same word level aligned parallel corpus as the one in Section 3.1. The fact that we only have a handful of parameters to fit eased the data sparseness problem.

3.4 N-Gram based Grammar Learner

Observing the success of Phrase based models, we built a second grammar learner that focuses on treelets that are n-gram phrase. This grammar learner extracts all the corresponding n-grams from the parallel trees if they are treelets on both sides. Formally, a pair of treelets ET_e and ET_f would be extracted if the following conditions suffice:

$$\left\{ (e_i, f_j) \middle| \begin{array}{l} e_i \notin ET_e,\ f_j \in ET_f \\ (e_i, f_j) \in alignment \end{array} \right\} = \Phi \qquad (6.1)$$

and

$$\left\{ (e_i, f_j) \middle| \begin{array}{l} e_i \in ET_e,\ f_j \notin ET_f \\ (e_i, f_j) \in alignment \end{array} \right\} = \Phi \qquad (6.2)$$

while ET_e and ET_f are both treelets that have n-grams as surface strings. For example, in Figure 4, the following are acceptable treelet pairs:

(The girl – The girl), (kissed – gave, a, kiss) …

On the other hand, (a kitty – to her) , while being n-grams on both sides, is not acceptable since the right hand side is not a treelet.

4 The Machine Translation System

4.1 System Architecture

As discussed before (see Figure 1 and 2), the architecture of our syntax based statistical MT system is illustrated in Figure 5. This is a non-deterministic process. The MT decoding starts first by decomposing the input dependency tree in to elementary trees. Each decomposition is indeed a derivation process on the foreign side of PSDIG. Then the elementary trees go through a transfer phase and target ETs are combined together.

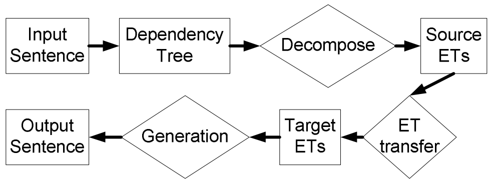


Figure 5. System architecture

4.2 The Original Model for PSDIG

The stochastic tree-to-tree transducer we propose models MT as a probabilistic optimization process.

Let f be the input sentence (foreign), and e be the output (English). The best translation is:

$$e^* = \arg\max_e \mathrm{P}(f \mid e)\mathrm{P}(e) \tag{7}$$

Assuming the decomposition of the foreign tree is given, our approach, which is based on ETs, uses the graphical model shown in Figure 6. The left side is the input dependency tree (foreign) and the right side is the output dependency tree (English). Each circle stands for an ET. The solid lines denote the syntactical dependencies while the dashed arrows denote the statistical dependencies.

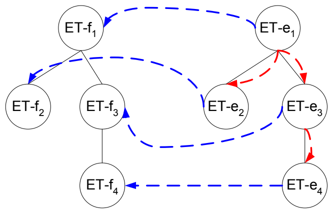


Figure 6 The graphical model

Let $\mathrm{T}(x)$ be the dependency tree constructed from sentence x. A tree-decomposition function $\mathrm{D}(t)$ is defined on a dependency tree t, and outputs a certain ET derivation tree of t, which is generated by decomposing t into ETs. Given t, there could be multiple decompositions. Conditioned on decomposition D, we can rewrite (7) as:

$$\begin{aligned} e^* &= \arg\max_e \sum_D \mathrm{P}(f, e \mid D)\mathrm{P}(D) \\ &= \arg\max_e \sum_D \mathrm{P}(f \mid e, D)\mathrm{P}(e \mid D)\mathrm{P}(D) \end{aligned} \tag{8}$$

By definition, the ET derivation trees of the input and output trees should be isomorphic: $\mathrm{D}(\mathrm{T}(f)) \cong \mathrm{D}(\mathrm{T}(e))$. Let $\mathrm{Tran}(u)$ be a set of possible translations for the ET u. We have:

$$\begin{aligned} \mathrm{P}(f \mid e, D) &= \mathrm{P}(\mathrm{T}(f) \mid \mathrm{P}(\mathrm{T}(e), D) \\ &= \prod_{u \in \mathrm{D}(\mathrm{T}(f)),\ v \in \mathrm{D}(\mathrm{T}(e)),\ v \in \mathrm{Tran}(u)} \mathrm{P}(u \mid v) \end{aligned} \tag{9}$$

For any ET v in a given ET derivation tree d, let $\mathrm{Root}(d)$ be the root ET of d, and let $\mathrm{Parent}(v)$ denote the parent ET of v. We have:

$$\begin{aligned} \mathrm{P}(e \mid D) &= \mathrm{P}(\mathrm{T}(e) \mid D) = \mathrm{P}\big(\mathrm{Root}\big(\mathrm{D}(\mathrm{T}(e))\big)\big) \cdot \\ &\cdot \left(\prod_{v \in \mathrm{D}(\mathrm{T}(e)), v \neq \mathrm{Root}(\mathrm{D}(\mathrm{T}(e)))} \mathrm{P}(v \mid \mathrm{Parent}(v)) \right) \end{aligned} \tag{10}$$

$$\begin{aligned} \mathrm{P}\big(v \mid \mathrm{Parent}(v)\big) &= \mathrm{P}_{lex}\big(v \mid \mathrm{Parent}(v)\big) \cdot \\ &\cdot \mathrm{P}_{sync}\big(v \mid \mathrm{Parent}(v)\big) \cdot \mathrm{P}_{direction}\big(v \mid \mathrm{Parent}(v)\big) \end{aligned} \tag{11}$$

Due to space limitations, further details of (11) are not discussed in this paper.

The prior probability of tree decomposition is defined as: $\mathrm{P}\big(\mathrm{D}(\mathrm{T}(f))\big) = \prod_{u \in \mathrm{D}(\mathrm{T}(f))} \mathrm{P}(u)$ (12)

For efficiency reasons, we use maximum approximation for (3). Instead of summing over all the possible decompositions, we only search for the best decomposition as follows:

$$e^*, D^* = \arg\max_{e,D} \mathrm{P}(f \mid e, D)\mathrm{P}(e \mid D)\mathrm{P}(D) \tag{13}$$

So bringing equations (4) to (9) together, the best translation would maximize:

$$\prod \mathrm{P}(u \mid v) \cdot \mathrm{P}\big(\mathrm{Root}(e)\big) \cdot \left(\prod \mathrm{P}(v \mid \mathrm{Parent}(v)) \right) \cdot \prod \mathrm{P}(u) \tag{14}$$

We refer to (14) as "the original model".

4.3 Interpolating the Models

As discussed in (Och, 2003), interpolating several probabilistic models in a log-linear fashion enhances the MT system performance. So, in addition to the original probabilistic model defined for PSDIG, we want to add the following models:

- A tri-gram language model using modified Kneser-Ney smoothing (Chen and Goodman, 1998). We refer to the tri-gram language model as $\mathrm{P}_{trigram}(e)$.
- Word count of the output sentence l to inhibit too short outputs, also called length penalty.
- Similarly, we add IBM Model 1 for both directions to optimize lexical choices, we call these two models $\mathrm{P}_m(f \mid e)$ and $\mathrm{P}_m(e \mid f)$.
- Lastly, it is also desirable to have (14) in an opposite direction (the foreign side generating the English side). We added $\mathrm{P}(e \mid f, D)$.

Hence, the best translation according to the interpolated model will maximize:

$$\begin{aligned} &\mathrm{P}(f \mid e, D)^{\lambda_{fe}} \cdot \mathrm{P}(e \mid f, D)^{\lambda_{ef}} \cdot \mathrm{P}(D)^{\lambda_D} \cdot \mathrm{P}(e \mid D)^{\lambda_{dep}} \\ &\cdot \mathrm{P}_m(f \mid e)^{\lambda_{mfe}} \cdot \mathrm{P}_m(e \mid f)^{\lambda_{mef}} \cdot \mathrm{P}_{trigram}(e)^{\lambda_{trigram}} \cdot l^{\lambda_{dw}} \end{aligned} \tag{15}$$

4.4 Greedy Decoding

It was shown in (Ding and Palmer, 2005) that for the original model of PSDIG, as in (14), a linear time decoding algorithm can be found, excluding the parsing time for the input string. With the addition of a trigram language model, the ETs are globally coupled together and the conditions for dynamic programming no longer hold. Hence, we use greedy search for decoding.

The decoder starts with one possible output s_0, and randomly changes the choices made for the ET transfer process, and a resulting solution s_1 is generated. s_1 is accepted if the score (15) improves and is rejected otherwise. Same is true for transitioning from solution s_i to solution s_{i+1}.

Interestingly, the dynamic programming decoder which only optimizes (14) can be used as a convenient starting point of the greedy decoder.

5 Evaluation

We implemented the above approach for a Chinese-English machine translation system. We used an automatic dependency parser (McDonald et al., 2005), trained using the Penn English/Chinese Treebanks. The training set consists of Xinhua newswire data from LDC and the FBIS data, total words: 7.4M English + 6.2M Chinese. The language model is trained using the Xinhua portion of the Gigaword corpus (30.0M words) with modified Kneser-Ney smoothing. The MT systems were evaluated using the *n*-gram based Bleu (Papineni et al., 2002), configured as case insensitive.

We compared to two systems: IBM Model 4 (Brown et al., 1993, Germann et al., 2001) and phrased based SMT system Pharaoh (Koehn, 2004).

Following (Och, 2003), we used the development test data from the 2001 NIST MT evaluation workshop to run error minimization training (206 sentences, 4 references each, 5945 words).

We used the oracle score for the top 100 translations as the measure for the potential of possible discriminative training. The "oracle" translations are picked by comparing with the references.

Model 4	PSDIG	PSDIG Top-100 Oracle	Pharaoh	Pharaoh Top-100 Oracle
13.1	30.6	36.8	30.8	34.7

Table 1. Results on NIST 2001 devtest

The system in (Ding and Palmer, 2005) achieves Bleu score 14.5. Hence the improved techniques in this paper doubled the bleu score of the system.

The current results for the PSDIG system is achieved by merging the rules learned using both the exhaustive learner and the n-gram learner, the details of the two learners are given below:

	Exhaustive	N-gram	Merged
Bleu	27.9	25.7	30.6
# rules	1.8M	0.7M	2.3M

Table 2. Merging the rules

As shown above, merging the rules learned by both grammar learners improved system performance. Interestingly, only a small percentage of the resultant rules from the two learners overlaps. The Pharaoh system has 3.3M phrase translation rules.

We further compared the systems using the Xinhua portion of the NIST MT evaluation 2003 test set (424 sentences, 4 references each, 10731 words). Results are shown below:

Model 4	PSDIG	PSDIG Top-100 Oracle	Pharaoh	Pharaoh Top-100 Oracle
12.3	23.1	28.8	23.0	28.8

Table 3. Results on NIST 2003 Xinhua portion

It is interesting to compare PSDIG outputs and the Pharaoh outputs. To do so we computed the oracle of merging the two outputs together. More specifically, we calculated the oracle score for merging the top-1 outputs and top-100 outputs of the two systems. To put the scores in context, we also calculated the bleu score for human translations. (Each human translation is evaluated using the rest 3 as references, and the results are averaged).

The result we got on the same two datasets mentioned above are as follows:

	PSDIG + Pharaoh Top 1 Oracle	PSDIG + Pharaoh Top-100 Oracle	Human (3 refs average)
Bleu	35.0	40.7	37.4

Table 4. Oracles of merging the two systems on NIST 2001 devtest data

	PSDIG + Pharaoh Top 1 Oracle	PSDIG + Pharaoh Top-100 Oracle	Human (3 refs average)
Bleu	27.7	32.9	37.0

Table 5. Oracles of merging the two systems on NIST 2003 Xinhua portion

Comparing with single system scores, we conclude that PSDIG and Pharaoh each excel on different sentences. The oracle of merging the two systems points to the possibility of future work in combining phrase-based MT and PSDIG together.

6 Discussions

By conducting a qualitative error analysis, we found that the PSDIG system performance is hurt mainly by the following reasons:

- Parsing errors result in "broken dependencies": Suppose for the English phrase "24.7 per cent", instead of (24.7 (per (cent))), the parser parsed it as (24.7) (per (cent)), i.e. two separate dependency sub-trees. If the Chinese side it is written as one token "bai-fen-zhi-24.7" (in pinyin form), the grammar learner cannot learn this rule correctly. This does not affect the phrase system, however.

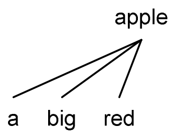


Figure 7. Dependency tree for "a big red apple"

- Currently an ET in a dependency tree has to contain a root. Linguistically, it has to be a constituent minus some child constituents. An ET cannot be a constituent plus a constituent. For example, in Figure 7, for the structure ((a) (big) (red) apple), the PSDIG is able to learn how ((a) apple), ((a) (big) apple), ((a) (red) apple) or ((a) (big) (red) apple) are translated, but it currently cannot use ((a) (big) (red)) as an ET– since it does not have a root.
- Sometimes the input is fragmented, e.g. telegram heads. Fragmented words tend to be sprayed all over the English side, since each word has the freedom to move on the English side and movements near the top of the dependency tree may result very long distance movements in the surface string.

We intend to address the above issues by adding to the grammar learner the capability to handle more complicated root-unlexicalized ETs in the future, e.g. ((a) (big) (red) X).

One possible strength of the PSDIG system is that the dependency trees on both languages provide a richer set of features for stronger models to handle more sophisticated language phenomena, e.g. case consistency, number consistency, mechanical translation of numerical values, etc.

We believe the system performance can be further improved by introducing other grammar learners and better quality control of the learned treelets.

On the other hand, it is reported in (Charniak et al. 2003) that the Bleu evaluation metric tends to reward more local word choices rather than global accuracy. In our system, whether the incorporation of syntax for both source and target languages has provided additional advantages beyond what is measured by Bleu needs further investigation.

7 Conclusions and Future Work

In this paper our work is based on a syntax-based statistical MT system using Probabilistic Synchronous Dependency Insertion Grammars. We improved the system performance by introducing two new grammar learners and a new decoder that incorporates several models, including a tri-gram language model.

Future work includes a learner to induce the more complicated root-unlexicalized ETs for the PSDIG and possibly stronger models to handle more sophisticated language phenomena. Another possibility is to combine the outputs of a phrase based MT system and PSDIG together.

References

H. Alshawi, S. Bangalore, S. Douglas. 2000. Learning dependency translation models as collections of finite state head transducers. *Comp. Linguistics*, 26(1):45-60.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2): 263-311.

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based Language Models for Machine Translation. In Proc. MT Summit IX.

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

Michael John Collins. 1999. Head-driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania, Philadelphia.

Ding and Palmer. 2005. Machine Translation Using Probabilistic Synchronous Dependency Insertion Grammars. In ACL-05.

Bonnie J. Dorr. 1994. Machine translation divergences: A formal description and proposed solution. *Computational Linguistics*, 20(4): 597-633.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In ACL-03.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP-02*.

Ulrich Germann, Michael Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2001. Fast Decoding and Optimal Decoding for Machine Translation. ACL-01.

Daniel Gildea. 2003. Loosely tree based alignment for machine translation. ACL-03, Japan.

Jonathan Graehl and Kevin Knight. 2004. Training Tree Transducers. In NAACL/HLT-2004

Jan Hajic, et al. 2002. Natural language generation in the context of MT. Summer workshop final report, CLSP, Johns Hopkins University, Baltimore.

Dekang Lin, 2004. A Path-based Transfer Model for Machine Translation. Coling-04.

Rebecca Hwa, Philip S. Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. ACL-02

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In Proc. of the 6th AMTA, pages 115-124.

Dan Melamed. 2003. Multitext Grammars and Synchronous Parsers, In NAACL/HLT-2003.

Ryan McDonald, Koby Crammer and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers. ACL-05.

Franz Josef Och and Hermann Ney. 2002. Discriminativetraining and maximum entropy models for statisticalmachine translation. ACL-02.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. ACL-03.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation, Computational Linguistics, 30:417-449.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. ACL-02, Philadelphia, USA.

Chris Quirk, Arul Menezes and Colin Cherry. 2005. Dependency Treelet Translation: Syntactically Informed Phrasal SMT

S. M. Shieber and Y. Schabes. 1990. *Synchronous Tree-Adjoining Grammars*, Proc. of 13th COLING, 1990.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):3-403.

Kenji Yamada and Kevin Knight. 2001. A syntax based statistical translation model. ACL-01, France.