Synchronous Dependency Insertion Grammars
A Grammar Formalism for Syntax Based Statistical MT

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

Abstract

This paper introduces a grammar formalism specifically designed for syntax-based statistical machine translation. The synchronous grammar formalism we propose in this paper takes into consideration the pervasive structure divergence between languages, which many other synchronous grammars are unable to model. A Dependency Insertion Grammars (DIG) is a generative grammar formalism that captures word order phenomena within the dependency representation. Synchronous Dependency Insertion Grammars (SDIG) is the synchronous version of DIG which aims at capturing structural divergences across the languages. While both DIG and SDIG have comparatively simpler mathematical forms, we prove that DIG nevertheless has a generation capacity weakly equivalent to that of CFG. By making a comparison to TAG and Synchronous TAG, we show how such formalisms are linguistically motivated. We then introduce a probabilistic extension of SDIG. We finally evaluated our current implementation of a simplified version of SDIG for syntax based statistical machine translation.

1 Introduction

Dependency grammars have a long history and have played an important role in machine translation (MT). The early use of dependency structures in machine translation tasks mainly fall into the category of transfer based MT, where the dependency structure of the source language is first analyzed, then transferred to the target language by using a set of transduction rules or a transfer lexicon, and finally the linear form of the target language sentence is generated.

While the above approach seems to be plausible, the transfer process demands intense human effort in creating a working transduction rule set or a transfer lexicon, which largely limits the performance and application domain of the resultant machine translation system.

In the early 1990s, (Brown et. al. 1993) introduced the idea of statistical machine translation, where the word to word translation probabilities and sentence reordering probabilities are estimated from a large set of parallel sentence pairs. By having the advantage of leveraging large parallel corpora, the statistical MT approach outperforms the traditional transfer based approaches in tasks for which adequate parallel corpora is available (Och, 2003). However, a major criticism of this approach is that it is void of any internal representation for syntax or semantics.

In recent years, hybrid approaches, which aim at applying statistical learning to structured data, began to emerge. Syntax based statistical MT approaches began with (Wu 1997), who introduced a polynomial-time solution for the alignment problem based on synchronous binary trees. (Alshawi et al., 2000) extended the tree-based approach by representing each production in parallel dependency trees as a finite-state transducer. (Yamada and Knight, 2001, 2002) model translation as a sequence of operations transforming a syntactic tree in one language into the string of the second language.

The syntax based statistical approaches have been faced with the major problem of pervasive structural divergence between languages, due to both systematic differences between languages (Dorr, 1994) and the vagaries of loose translations in real corpora. While we would like to use syntactic information in both languages, the problem of non-isomorphism grows when trees in both languages are required to match.

To allow the syntax based machine translation approaches to work as a generative process, certain isomorphism assumptions have to be made. Hence a reasonable question to ask is: to what extent should the grammar formalism, which we choose to represent syntactic language transfer, assume isomorphism between the structures of the two languages?

(Hajic et al., 2002) allows for limited non-isomorphism in that n-to-m matching of nodes in the two trees is permitted. However, even after extending this model 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 using one input tree than with a tree-to-tree model using two.

At the same time, grammar theoreticians have proposed various generative synchronous grammar formalisms for MT, such as Synchronous Context Free Grammars (S-CFG) (Wu, 1997) or Synchronous Tree Adjoining Grammars (S-TAG) (Shieber and Schabes, 1990). Mathematically, generative synchronous grammars share many good properties similar to their monolingual counterparts such as CFG or TAG (Joshi and Schabes, 1992). If such a synchronous grammar could be learnt from parallel corpora, the MT task would become a mathematically clean generative process.

However, the problem of inducing a synchronous grammar from empirical data was never solved. For example, Synchronous TAGs, proposed by (Shieber and Schabes, 1990), which were introduced primarily for semantics but were later also proposed for translation. From a formal perspective, Syn-TAGs characterize the correspondences between languages by a set of synchronous elementary tree pairs. While examples show that this formalism does capture certain cross language structural divergences, there is not, to our knowledge, any successful statistical learning method to learn such a grammar from empirical data. We believe that this is due to the limited ability of Synchronous TAG to model structure divergences. This observation will be discussed later in Section 5.

We studied the problem of learning synchronous syntactic sub-structures (parallel dependency treelets) from unaligned parallel corpora in (Ding and Palmer, 2004). At the same time, we would like to formalize a synchronous grammar for syntax based statistical MT. The necessity of a well-defined formalism and certain limitations of the current existing formalisms, motivate us to design a new synchronous grammar formalism which will have the following properties:

1. Linguistically motivated: it should be able to capture most language phenomena, e.g. complicated word orders such as “wh” movement.
2. Without the unrealistic word-to-word isomorphism assumption: it should be able to capture structural variations between the languages.
3. Mathematically rigorous: it should have a well defined formalism and a proven generation capacity, preferably context free or mildly context sensitive.
4. Generative: it should be “generative” in a mathematical sense. This property is essential for the grammar to be used in statistical MT. Each production rule should have its own probability, which will allow us to decompose the overall translation probability.
5. Simple: it should have a minimal number of different structures and operations so that it will be learnable from the empirical data.

In the following sections of this paper, we introduce a grammar formalism that satisfies the above properties: Synchronous Dependency Insertion Grammar (SDIG). Section 2 gives an informal look at the desired capabilities of a monolingual version Dependency Insertion Grammar (DIG) by addressing the problems with previous dependency grammars. Section 3 gives the formal definition of the DIG and shows that it is weakly equivalent to Context Free Grammar (CFG). Section 4 shows how DIG is linguistically motivated by making a comparison between DIG and Tree Adjoining Grammar (TAG). Section 5 specifies the Synchronous DIG and Section 6 gives the probabilistic extension of SDIG.

2 Issues with Dependency Grammars

2.1 Dependency Grammars and Statistical MT

According to (Fox, 2002), dependency representations have the best phrasal cohesion properties across languages. The percentage of head crossings per chance is 12.62% and that of modifier crossings per chance is 9.22%. Observing this fact, it is reasonable to propose a formalism that handles language transfer based on dependency structures.

What is more, if a formalism based on dependency structures is made possible, it will have the nice property of being simple, as expressed in the following table:

	CFG	TAG	DG
Node#	*2n*	*2n*	*n*
Lexicalized?	NO	YES	YES
Node types	2	2	1*
Operation types	1	2	1*

(*: will be shown later in this paper)

Figure 1.

The simplicity of a grammar is very important for statistical modeling, i.e. when it is being learned from the corpora and when it is being used in machine translation decoding, we don’t need to condition the probabilities on two different node types or operations.

At the same time, dependency grammars are inherently lexicalized in that each node is one word. Statistical parsers (Collins 1999) showed performance improvement by using bilexical probabilities, i.e. probabilities of word pair occurrences. This is what dependency grammars model explicitly.

2.2 A Generative Grammar?

Why do we want the grammar for statistical MT to be generative? First of all, generative models have long been studied in the machine learning community, which will provide us with mathematically rigorous algorithms for training and decoding. Second, CFG, the most popular formalism in describing natural language phenomena, is generative. Certain ideas and algorithms can be borrowed from CFG if we make the formalism generative.

While there has been much previous work in formalizing dependency grammars and in its application to the parsing task, until recently (Joshi and Rambow, 2003), little attention has been given to the issue of making the proposed dependency grammar generative. And in machine translation tasks, although using dependency structures is an old idea, little effort has been made to propose a formal grammar which views the composition and decomposition of dependency trees as a generative process from a formal perspective.

There are two reasons for this fact: (1) The "pure" dependency trees do not have nonterminals. The standard solution to this problem was introduced as early as (Gaifman 1965), where he proposed adding syntactic categories to each node on the dependency tree. (2) However, there is a deeper problem with dependency grammar formalisms, as observed by (Rambow and Joshi 1997). In the dependency representation, it is hard to handle complex word order phenomena without resorting to global word order rules, which makes the grammar no longer generative. This will be explored in the next subsection (2.3).

2.3 Non-projectivity

Non-projectivity has long been a major obstacle for anyone who wants to formalize dependency grammar. When we draw projection lines from the nodes in the dependency trees to a linear representation of the sentence, if we cannot do so without having one or more projection lines going across at least one of the arcs of the dependency tree, we say the dependency tree is non-projective.

A typical example for non-projectivity is "wh" movement, which is illustrated below.

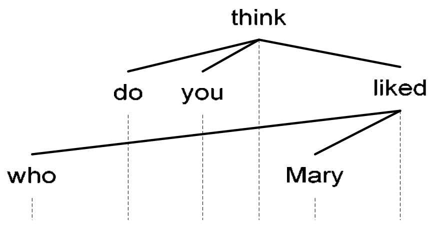

Figure 2.

Our solution for this problem is given in section 4 and in the next section we will first give the formal definition of the monolingual Dependency Insertion Grammar.

3 The DIG Formalism

3.1 Elementary Trees

Formally, the Dependency Insertion Grammar is defined as a six tuple (C, L, A, B, S, R). C is a set of syntactic categories and L is a set of lexical items. A is a set of Type-A trees and B is a set of Type-B trees (defined later). S is a set of the starting categories of the sentences. R is a set of word order rules local to each node of the trees.

Each node in the DIG has three fields:

A Node consists of:
1. One lexical item
2. One corresponding category
3. One local word order rule.

We define two types of elementary trees in DIG: Type-A trees and Type-B trees. Both types of trees have one or more nodes. One of the nodes in an elementary tree is designated as the head of the elementary tree.

Type-A trees are also called "root lexicalized trees". They roughly correspond to the α trees in TAG. Type-A trees have the following properties:

Properties of a Type-A elementary tree:
1. The root is lexicalized.
2. The root is designated as the head of the tree
3. Any lexicalized node can take a set of unlexicalized nodes as its arguments.
4. The local word order rule specifies the relative order between the current node and all its immediate children, including the unlexicalized arguments.

Here is an example of a Type-A elementary tree for the verb "*like*". Note that the head node is marked with (@).

Please note that the placement of the dependency arcs reflects the relative order between the parent and all its immediate children.

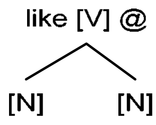

Figure 3

Type-B trees are also called "root unlexicalized trees". They roughly correspond to β trees in TAG and have the following properties:

Properties of a Type-B elementary tree:

1. The root is the ONLY unlexicalized node
2. One of the lexicalized nodes is designated as the head of the tree
3. Similar to Type-A trees, each node also have a word order rule that specifies the relative order between the current node and all its immediate children.

Here is and example of a Type-B elementary tree for the adverb "*really*"

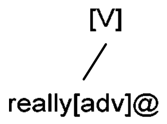

Figure 4

3.2 The Unification Operation

We define only one type of operation: unification for any DIG derivation:

Unification Operation:

When an unlexicalized node and a head node have the same categories, they can be merged into one node.

This specifies that an unlexicalized node cannot be unified with a non-head node, which guarantees limited complexity when a unification operation takes place.

After unification,

1. If the resulting tree is a Type-A tree, its root becomes the new root;
2. If the resulting tree is a Type-B tree, the root node involved in the unification operation becomes the new root.

Here is one example for the unification operation which adjoins the adverb "*really*" to the verb "*like*":

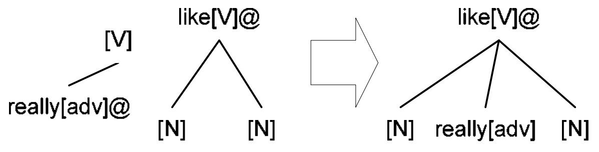

Figure 5

Note that for the above unification operation the dependency tree on the right hand side is just one of the possible resultant dependency trees. The strings generated by the set of possible resultant dependency trees should all be viewed as the language $L(DIG)$ generated by the DIG grammar.

Also note that the definition of DIG is preserved through the unification operation, as we have:

1. (Type-A) (unify) (Type A) = (Type-A)
2. (Type-A) (unify) (Type B) = (Type-A)
3. (Type-B) (unify) (Type B) = (Type-B)

3.3 Comparison to Other Approaches

There are two major differences between our dependency grammar formalism and that of (Joshi and Rambow, 2003):

1. We only define one unification operation, whereas (Joshi and Rambow, 2003) defined two operations: substitution and adjunction.
2. We introduce the concept of "heads" in the DIG so that the derivation complexity is significantly smaller.

3.4 Proof of Weak Equivalence between DIG and CFG

We prove the weak equivalence between DIG and CFG by first showing that the language that a DIG generates is a subset of one that a CFG generates, i.e. $L(DIG) \subseteq L(CFG)$. And then we show the opposite is also true: $L(CFG) \subseteq L(DIG)$.

3.4.1 $L(DIG) \subseteq L(CFG)$

The proof is given constructively. First, for each Type-A tree, we "insert" a "waiting for Type-B tree" argument at each possible slot underneath it with the category B. This process is shown below:

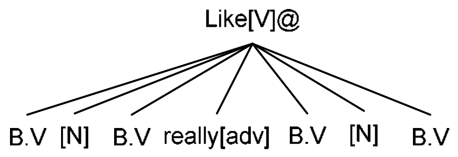

Figure 6

Then we "flatten" the Type-A tree to its linear form according to the local word order rule, which decides the relative ordering between the parent and all its children at each of the nodes. And we get:

$$NT\{A.C_H\} \rightarrow NT\{B.C_H\} w_0 NT\{C_0\} w_1 NT\{B.C_H\} \cdots$$
$$\cdots w_i NT\{C_j\} \cdots w_n NT\{B.C_H\}$$

- $w_0 \cdots w_n$ is the strings of lexical items
- $NT\{A.C_H\}$ is the nonterminal created for this Type-A tree, and C_H is the category of the head (root).
- $NT\{C_j\}$ is the nonterminal for each category
- $NT\{B.C_H\}$ is the nonterminal for each "Type-B site"

Similarly, for each Type-B tree we can create "Type-B site" under its head node. So we have:

$$NT\{RB.C_R\} \rightarrow w_0 NT\{B.C_H\} \cdots w_i \cdots NT\{B.C_H\} w_n$$

Then we create the production to take arguments:

$$NT\{C\} \rightarrow NT\{A.C\}$$

And the production rules to take Type-B trees:

$$NT\{B.C\} \rightarrow NT\{RB.C\} NT\{B.C\}$$
$$NT\{B.C\} \rightarrow NT\{B.C\} NT\{RB.C\}$$

Hence, a DIG can be converted to a CFG.

3.4.2 $L(CFG) \subseteq L(DIG)$

It is known that a context free grammar can be converted to Greibach Normal Form, where each production will have the form:

$A \rightarrow aV^*$, where V is the set of nonterminals

We simply construct a corresponding Type-A dependency tree as follows:

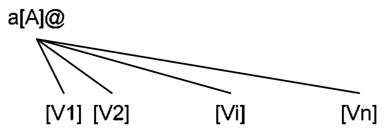

Figure 7

4 Compare DIG to TAG

A Tree Adjoining Grammars is defined as a five tuple (Σ, NT, I, A, S), where Σ is a set of terminals, NT is a set of nonterminals, I is a finite set of finite initial trees (α trees), A is a finite set of auxiliary trees (β trees), and S is a set of starting symbols. The TAG formalism defines two operations, substitution and adjunction.

A TAG derives a phrase-structure tree, called the "derived tree" and at the same time, in each step of the derivation process, two elementary trees are connected through either the substitution or adjunction operation. Hence, we have a "derivation tree" which represents the syntactic and/or logical relation between the elementary trees. Since each elementary tree of TAG has exactly one lexical node, we can view the derivation tree as a "Deep Syntactic Representation" (DSynR). This representation closely resembles the dependency structure of the sentence.

Here we show how DIG models different operations of TAG and hence handles word order phenomena gracefully.

We categorize the TAG operations into three different types: substitution, non-predicative adjunction and predicative adjunction.

- Substitution

We model the TAG substitution operation by having the embedded tree replaces the non-terminal that is in accordance with its root. An example for this type is the substitution of NP.

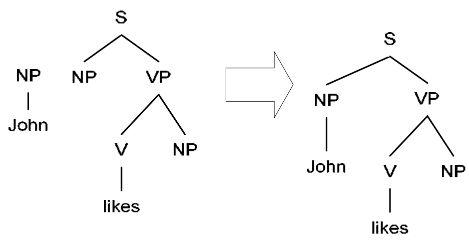

Figure 8a Substitution in TAG

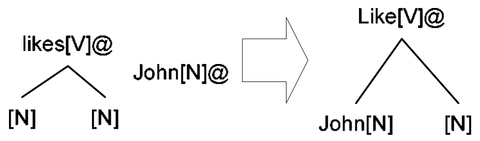

Figure 8b Substitution through DIG unification

- Non-predicative Adjunction

In TAG, this type of operation includes all adjunctions when the embedded tree does not contain a predicate, i.e. the root of the embedded tree is not an S. For example, the trees for adverbs are with root VP and are adjoined to non-terminal VPs in the matrix tree.

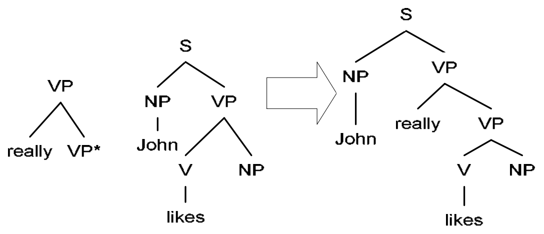

Figure 9a Non-predicative Adjunction in TAG

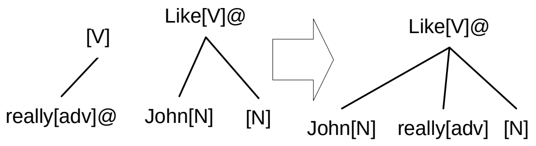

Figure 9b Non-predicative Adjunction through DIG unification

- Predicative Adjunction

This type of operation adjoins an embedded tree which contains a predicate, i.e. with a root S, to the matrix tree. A typical example is the sentence: *Who does John think Mary likes?*

This example is non-projective and has "wh" movement. In the TAG sense, the tree for "*does John think*" is adjoined to the matrix tree for "*Who Mary likes*". This category of operation has some interesting properties. The dependency relation of the embedded tree and the matrix tree is inverted. This means that if tree T1 is adjoined to T2, in non-predicative adjunction, T1 depends on T2, but in predicative adjunction, T2 depends on T1. In the above example, the tree with "*like*" depends on the tree with "*think*".

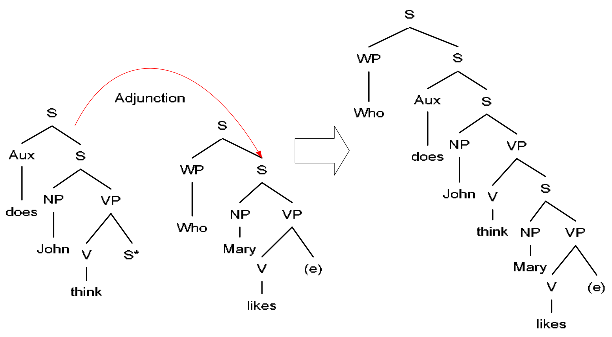

Figure 10a "Wh" movement through TAG (predicative) adjunction operation

Our solution is quite simple: when we are constructing the grammar, we invert the arc that points to a predicative clause. Despite the fact that the resulting dependency trees have certain arcs inverted, we will still be able to use localized word order rules and derive the desired sentence with the simple unification operation. As shown below:

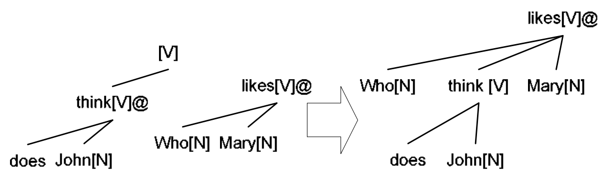

Figure 10b "Wh" movement through unification

Since TAG is mildly context sensitive, and we have shown in Section 3 that DIG is context free, we are not claiming the two grammars are weakly or strongly equivalent. Also, please note DIG does not handle all the non-projectivity issues due to its CFG equivalent generation capacity.

5 Synchronous DIG

5.1 Definition

(Wu, 1997) introduced synchronous binary trees and (Shieber, 1990) introduced synchronous tree adjoining grammars, both of which view the translation process as a synchronous derivation process of parallel trees. Similarly, with our DIG formalism, we can construct a Synchronous DIG by synchronizing both structures and operations in both languages and ensuring synchronous derivations.

Properties of SDIG:

1. The roots of both trees of the source and target languages are aligned, and have the same category
2. All the unlexicalized nodes of both trees are aligned and have the same category.
3. The two heads of both trees are aligned and have the same category.

Synchronous Unification Operation:

By the above properties of SDIG, we can show that unification operations are synchronized in both languages. Hence we can have synchronous unification operations.

5.2 Isomorphism Assumption

So how is SDIG different from other synchronous grammar formalisms?

As we know, a synchronous grammar derives both source and target languages through a series of synchronous derivation steps. For any tree-based synchronous grammar, the synchronous derivation would create two derivation trees for both languages which have isomorphic structure. Thus a synchronous grammar assumes certain isomorphism between the two languages which we refer to as the "isomorphism assumption".

Now we examine the isomorphism assumptions in S-CFG and S-TAG:

- For S-CFG, the substitutions for all the nonterminals need to be synchronous. Hence the isomorphism assumption for S-CFG is *isomorphic phrasal structure.*
- For S-TAG, all the substitution and adjunction operations need to be synchronous, and the derivation trees of both languages are isomorphic. The derivation tree for TAG is roughly equivalent to a dependency tree. Hence the isomorphism assumption for S-TAG is an *isomorphic dependency structure.*

As shown by real translation tasks, both of those assumptions would fail due to structural divergences between languages.

On the other hand SDIG does NOT assume word level isomorphism or isomorphic dependency trees. Since in the SDIG sense, the parallel dependency trees are in fact the "derived" form rather than the "derivation" form. In other words, SDIG assumes the isomorphism lies deeper than the dependency structure. It is "*the derivation tree of DIG*" that is isomorphic.

The following "pseudo-translation" example illustrates how SDIG captures structural divergence between the languages. Suppose we want to translate:

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

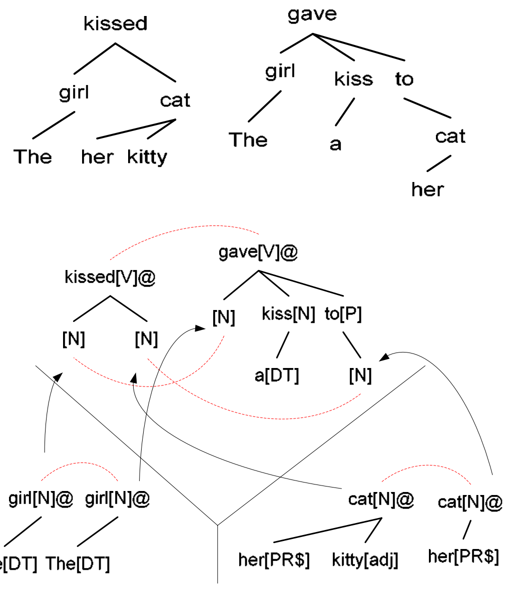

Figure 11

Note that both S-CFG and S-TAG won't be able to handle such structural divergence. However, when we view each of the two sentences as derived from three elementary trees in DIG, we can have a synchronous derivation, as shown below:

6 The Probabilistic Extension to SDIG and Statistical MT

The major reason to construct an SDIG is to have a generative model for syntax based statistical MT. By relying on the assumption that the derivation tree of DIG represents the probability dependency graph, we can build a graphical model which captures the following two statistical dependencies:

1. Probabilities of Elementary Tree unification (in the target language)
2. Probabilities of Elementary Tree transfer (between languages), i.e. the probability of two elementary trees being paired

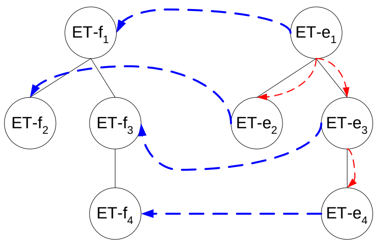

Figure 12

The above graph shows two isomorphic derivation trees for two languages. ET stands for elementary trees and dotted arcs denote the conditional dependence assumptions). Under the above model, the best translation is: $e^* = \arg\max_e P(f \mid e)P(e)$;

And $P(f \mid e) = \prod_i P(ET(f)_i \mid ET(e)_i)$; also we have $P(e) = \prod_i P\big(ET(e)_i \mid Parent(ET(e)_i)\big)$.

Hence, we can have PSDIG (probabilistic synchronous Dependency Insertion Grammar). Given the dynamic programming property of the above graphical model, an efficient polynomial time Viterbi decoding algorithm can be constructed.

7 Current Implementation

To test our idea, we implemented the above synchronous grammar formalism in a Chinese-English machine translation system. The actual implementation of the synchronous grammar used in the system is a scaled-down version of the SDIG introduced above, where all the word categories are treated as one. The reason for this simplification is that word category mappings across languages are not straightforward. Defining the word categories so that they can be consistent between the languages is a major goal for our future research.

The uni-category version of the SDIG is induced using the algorithm in (Ding and Palmer, 2004), which is a statistical approach to extracting parallel dependency structures from large scale parallel corpora. An example is given in Figure 12. We can construct the parallel dependency trees as shown in Figure 13a. The expected output of the above approach is shown in Figure 13b. (e) stands for an empty node trace.

- [*English*] *I have been here since 1947.*
- [*Chinese*] *Wo 1947 nian yilai yizhi zhu zai zheli.*

Wo	*1947*	*nian*	*yilai*	*yizhi*	*zhu*	*zai*	*zheli.*
I		*year*	*since*	*always*	*live*	*in*	*here*

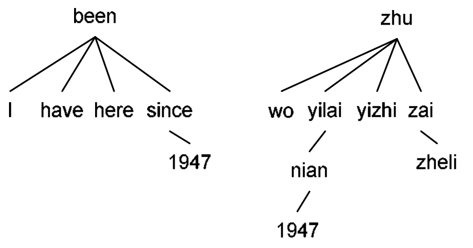

Figure 13a. Input

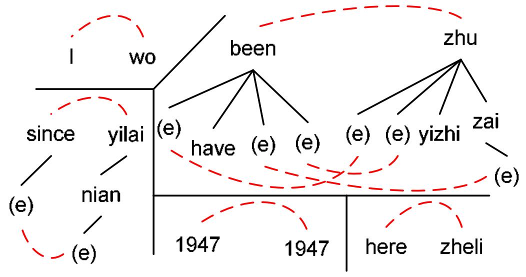

Figure 13b. Output (5 parallel elementary tree pairs)

We build a decoder for the model in Section 6 for our machine translation system. The decoder is based on a polynomial time decoding algorithm for fast non-isomorphic tree-to-tree transduction (Unpublished by the time of this paper).

We use an automatic syntactic parser (Collins, 1999; Bikel, 2002) to produce the parallel unaligned syntactic structures. The parser was trained using the Penn English/Chinese Treebanks. We then used the algorithm in (Xia 2001) to convert the phrasal structure trees into dependency trees.

The following table shows the statistics of the datasets we used. (Genre, number of sentence pairs, number of Chinese/English words, type and usage).

Dataset	Xinhua	FBIS	NIST
Genre	News	News	News
Sent#	56263	21003	206
Chn W#	1456495	522953	26.3 average
Eng W#	1490498	658478	32.5 average
Type	unaligned	unaligned	multi-reference
Usage	training	training	testing

Figure 14

The training set consists of Xinhua newswire data from LDC and the FBIS data. We filtered both datasets to ensure parallel sentence pair quality. We used the development test data from the 2001 NIST MT evaluation workshop as our test data for the MT system performance. In the testing data, each input Chinese sentence has 4 English translations as references, so that the result of the MT system can be evaluated using Bleu and NIST machine translation evaluation software.

	1-gram	2-gram	3-gram	4-gram
NIST:	4.3753	4.9773	5.0579	5.0791
BLEU:	0.5926	0.3417	0.2060	0.1353

Figure 15

The above table shows the cumulative Bleu and NIST n-gram scores for our current implementation; with the final Bleu score 0.1353 with average input sentence length of 26.3 words.

In comparison, in (Yamada and Knight, 2002), which was a phrasal structure based statistical MT system for Chinese to English translation, the Bleu score reported for short sentences (less than 14 words) is 0.099 to 0.102.

Please note that the Bleu/NIST scorers, while based on n-gram matching, do not model syntax during evaluation, which means a direct comparison between a syntax based MT system and a string based statistical MT system using the above scorer would favor the string based systems.

We believe that our results can be improved using a more sophisticated machine translation pipeline which has separate components that handle specific language phenomena such as named entities. Larger training corpora can also be helpful.

8 Conclusion

Finally, let us review whether the proposed SDIG formalism has achieved the goals we setup in Section 1 of this paper for a grammar formalism for Statistical MT applications:

1. Linguistically motivated: DIG captures word-order phenomena within the CFG domain.
2. SDIG dropped the unrealistic word-to-word isomorphism assumption and is able to capture structural divergences.
3. DIG is weakly equivalent to CFG.
4. DIG and SDIG are generative grammars.
5. They have both simple formalisms, only one type of node, and one type of operation.

9 Future Work

We observe from our testing results that the current simplified uni-category version of SDIG suffers from various grammatical errors, both in grammar induction and decoding, therefore our future work should focus on word category consistency between the languages so that a full-fledged version of SDIG can be used.

10 Acknowledgements

Our thanks go Aravind Joshi, Owen Rambow, Dekai Wu and all the anonymous reviewers of the previous versions of the paper, who gave us invaluable advices, suggestions and feedbacks.

References

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 2000. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26(1): 45-60.

Daniel M. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of HLT 2002*.

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

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

Yuan Ding and Martha Palmer. 2004. Automatic Learning of Parallel Dependency Treelet Pairs, in *Proceedings of The First International Joint Conference on Natural Language Processing* (IJCNLP-04).

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

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP-02*, pages 304-311

Daniel Gildea. 2003. Loosely tree based alignment for machine translation. In *Proceedings of ACL-03*

Jan Hajic, et al. 2002. Natural language generation in the context of machine translation. Summer workshop final report, Center for Language and Speech Processing, Johns Hopkins University, Baltimore.

Aravind Joshi and Owen Rambow. 2003. A formalism of dependency grammar based on Tree Adjoining Grammar. In *Proceedings of the first international conference on meaning text theory* (MTT 2003), June 2003.

Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, Tree Automata and Languages. Elsevier Science, 1992.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. *In Proceedings of* ACL-03), pages 160-167.

Owen Rambow and Aravind Joshi. 1997. A formal look at dependency grammars and phrase structures. In Leo Wanner, editor, *Recent Trends in Meaning-Text Theory*, pages 167-190.

S. M. Shieber and Y. Schabes. 1990. *Synchronous Tree-Adjoining Grammars*, Proceedings of the 13th COLING, pp. 253-258, August 1990.

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

Fei Xia. 2001. Automatic grammar generation from two different perspectives. Ph.D. thesis, University of Pennsylvania, Philadelphia.

Kenji Yamada and Kevin Knight. 2001. A syntax based statistical translation model. In *Proceedings of ACL-01*

Kenji Yamada and Kevin Knight. 2002. A decoder for syntax-based statistical MT. In *Proceedings of ACL-02*