

# A Systematic Comparison between Inversion Transduction Grammar and Linear Transduction Grammar for Word Alignment

Markus Saers and Joakim Nivre

Dept. of Linguistics & Philology  
Uppsala University

*first.last@lingfil.uu.se*

Dekai Wu

HKUST

Human Language Technology Center  
Dept. of Computer Science & Engineering  
Hong Kong Univ. of Science & Technology

*dekai@cs.ust.hk*

## Abstract

We present two contributions to grammar driven translation. First, since both Inversion Transduction Grammar and Linear Inversion Transduction Grammars have been shown to produce better alignments than the standard word alignment tool, we investigate how the trade-off between speed and end-to-end translation quality extends to the choice of grammar formalism. Second, we prove that Linear Transduction Grammars (LTGs) generate the same transductions as Linear Inversion Transduction Grammars, and present a scheme for arriving at LTGs by bilingualizing Linear Grammars. We also present a method for obtaining Inversion Transduction Grammars from Linear (Inversion) Transduction Grammars, which can speed up grammar induction from parallel corpora dramatically.

## 1 Introduction

In this paper we introduce Linear Transduction Grammars (LTGs), which are the bilingual case of Linear Grammars (LGs). We also show that LTGs are equal to Linear Inversion Transduction Grammars (Saers et al., 2010). To be able to induce transduction grammars directly from parallel corpora an approximate search for parses is needed. The trade-off between speed and end-to-end translation quality is investigated and compared to Inversion Transduction Grammars (Wu, 1997) and the standard tool for word alignment,

GIZA++ (Brown et al., 1993; Vogel et al., 1996; Och and Ney, 2003). A heuristic for converting stochastic bracketing LTGs into stochastic bracketing ITGs is presented, and fitted into the speed-quality trade-off.

In section 3 we give an overview of transduction grammars, introduce LTGs and show that they are equal to LITGs. In section 4 we give a short description of the rationale for the transduction grammar pruning used. In section 5 we describe a way of seeding a stochastic bracketing ITG with the rules and probabilities of a stochastic bracketing LTG. Section 6 describes the setup, and results are given in section 7. Finally, some conclusions are offered in section 8

## 2 Background

Any form of automatic translation that relies on generalizations of observed translations needs to align these translations on a sub-sentential level. The standard way of doing this is by aligning words, which works well for languages that use white space separators between words. The standard method is a combination of the family of IBM-models (Brown et al., 1993) and Hidden Markov Models (Vogel et al., 1996). These methods all arrive at a function ( $A$ ) from language 1 ( $F$ ) to language 2 ( $E$ ). By running the process in both directions, two functions can be estimated and then combined to form an alignment. The simplest of these combinations are intersection and union, but usually, the intersection is heuristically extended. Transduction grammars on the other hand, impose a shared structure on the sentence pairs, thus forcing a consistent alignment in both directions. This method

has proved successful in the settings it has been tried (Zhang et al., 2008; Saers and Wu, 2009; Haghghi et al., 2009; Saers et al., 2009; Saers et al., 2010). Most efforts focus on cutting down time complexity so that larger data sets than toy-examples can be processed.

### 3 Transduction Grammars

Transduction grammars were first introduced in Lewis and Stearns (1968), and further developed in Aho and Ullman (1972). The original notation called for regular CFG-rules in language  $F$  with rephrased  $E$  productions, either in curly brackets, or comma separated. The bilingual version of CFGs is called Syntax-Directed Transduction Grammars (SDTGs). To differentiate identical nonterminal symbols, indices were used (the bag of nonterminals for the two productions are equal by definition).

$$\begin{aligned} A &\rightarrow B^{(1)} a B^{(2)} \{x B^{(1)} B^{(2)}\} \\ &= A \rightarrow B^{(1)} a B^{(2)}, x B^{(1)} B^{(2)} \end{aligned}$$

The semantics of the rules is that one nonterminal rewrites into a bag of nonterminals that is distributed independently in the two languages, and interspersed with any number of terminal symbols in the respective languages. As with CFGs, the terminal symbols can be factored out into preterminals with the added twist that they are shared between the two languages, since preterminals are formally nonterminals. The above rule can thus be rephrased as

$$\begin{aligned} A &\rightarrow B^{(1)} X^{a/x} B^{(2)}, X^{a/x} B^{(1)} B^{(2)} \\ X^{a/x} &\rightarrow a, x \end{aligned}$$

In this way, rules producing nonterminals and rules producing terminals can be separated. Since only nonterminals are allowed to move, their movement can be represented as the original sequence of nonterminals and a permutation vector as follows:

$$\begin{aligned} A &\rightarrow B X^{a/x} B ; 1, 0, 2 \\ X^{a/x} &\rightarrow a, x \end{aligned}$$

To keep the reordering as monotone as possible, the terminals  $a$  and  $x$  can be produced separately,

but doing so eliminates any possibility of parameterizing their lexical relationship. Instead, the individual terminals are pair up with the empty string ( $\epsilon$ ).

$$\begin{aligned} A &\rightarrow X^x B X^a B ; 0, 1, 2, 3 \\ X^a &\rightarrow a, \epsilon \\ X^x &\rightarrow \epsilon, x \end{aligned}$$

Lexical rules involving the empty string are referred to as singletons. Whenever a preterminal is used to pair up two terminal symbols, we refer to that pair of terminals as a *biterminal*, which will be written as  $e/f$ .

Any SDTG can be rephrased to contain permuted nonterminal productions and biterminal productions only, and we will call this the normal form of SDTGs. Note that it is not possible to produce a two-normal form for SDTGs, as there are some rules that are not binarizable (Wu, 1997; Huang et al., 2009). This is an important point to make, since efficient parsing for CFGs is based on either restricting parsing to only handle binary grammars (Cocke, 1969; Kasami, 1965; Younger, 1967), or rely on on-the-fly binarization (Earley, 1970). When translating with a grammar, parsing only has to be done in  $F$ , which is binarizable (since it is a CFG), and can therefore be computed in polynomial time ( $\mathcal{O}(n^3)$ ). Once there is a parse tree for  $F$ , the corresponding tree for  $E$  can be easily constructed. When inducing a grammar from examples, however, biparsing (finding an analysis that is consistent across a sentence pair) is needed. The time complexity for biparsing with SDTGs is  $\mathcal{O}(n^{2n+2})$ , which is clearly intractable.

Inversion Transduction Grammars or ITGs (Wu, 1997) are transduction grammars that have a two-normal form, thus guaranteeing binarizability. Defining the rank of a rule as the number of nonterminals in the production, and the rank of a grammar as the highest ranking rule in the rule set, ITGs are *a)* any SDTG of rank two, *b)* any SDTG of rank three or *c)* any SDTG where no rule has a permutation vector other than identity permutation or inversion permutation. It follows from this definition that ITGs have a two-normal form, which is usually expressed as SDTG rules,

with brackets around the production to distinguish the different kinds of rules from each other.

$$\begin{aligned} A \rightarrow B C ; 0, 1 &= A \rightarrow [ B C ] \\ A \rightarrow B C ; 1, 0 &= A \rightarrow \langle B C \rangle \\ A \rightarrow e/f &= A \rightarrow e/f \end{aligned}$$

By guaranteeing binarizability, biparsing time complexity becomes  $\mathcal{O}(n^6)$ .

There is an even more restricted version of SDTGs called Simple Transduction Grammar (STG), where no permutation at all is allowed, which can also biparse a sentence pair in  $\mathcal{O}(n^6)$  time.

A Linear Transduction Grammar (LTG) is a bilingual version of a Linear Grammar (LG).

**Definition 1.** An LG in normal form is a tuple

$$\mathcal{G}_L = \langle N, \Sigma, R, S \rangle$$

Where  $N$  is a finite set of nonterminal symbols,  $\Sigma$  is a finite set of terminal symbols,  $R$  is a finite set of rules and  $S \in N$  is the designated start symbol. The rule set is constrained so that

$$R \subseteq N \times (\Sigma \cup \{\epsilon\})N(\Sigma \cup \{\epsilon\}) \cup \{\epsilon\}$$

Where  $\epsilon$  is the empty string.

To bilingualize a linear grammar, we will take the same approach as taken when a finite-state automaton is bilingualized into a finite-state transducer. That is: to replace all terminal symbols with biterminal symbols.

**Definition 2.** An LTG in normal form is a tuple

$$\mathcal{TG}_L = \langle N, \Sigma, \Delta, R, S \rangle$$

Where  $N$  is a finite set of nonterminal symbols,  $\Sigma$  is a finite set of terminal symbols in language  $E$ ,  $\Delta$  is a finite set of terminal symbols in language  $F$ ,  $R$  is a finite set of linear transduction rules and  $S \in N$  is the designated start symbol. The rule set is constrained so that

$$R \subseteq N \times \Psi N \Psi \cup \{\langle \epsilon, \epsilon \rangle\}$$

Where  $\Psi = \Sigma \cup \{\epsilon\} \times \Delta \cup \{\epsilon\}$  and  $\epsilon$  is the empty string.

Graphically, we will represent LTG rules as production rules with biterminals:

$$\begin{aligned} \langle A, \langle x, p \rangle B \langle y, q \rangle \rangle &= A \rightarrow x/p B y/q \\ \langle A, \langle \epsilon, \epsilon \rangle \rangle &= B \rightarrow \epsilon/\epsilon \end{aligned}$$

Like STGs, LTGs do not allow any reordering, and are monotone, but because they are linear, this has no impact on expressiveness, as we shall see later.

Linear Inversion Transduction Grammars (LITGs) were introduced in Saers et al. (2010), and represent ITGs that are allowed to have at most one nonterminal symbol in each production. These are attractive because they can biparse a sentence pair in  $\mathcal{O}(n^4)$  time, which can be further reduced to linear time by severely pruning the search space. This makes them tractable for large parallel corpora, and a viable way to induce transduction grammars from large parallel corpora.

**Definition 3.** An LITG in normal form is a tuple

$$\mathcal{TG}_{LI} = \langle N, \Sigma, \Delta, R, S \rangle$$

Where  $N$  is a finite set of nonterminal symbols,  $\Sigma$  is a finite set of terminal symbols from language  $E$ ,  $\Delta$  is a finite set of terminal symbols from language  $F$ ,  $R$  is a set of rules and  $S \in N$  is the designated start symbol. The rule set is constrained so that

$$R \subseteq N \times \{\square, \langle \rangle\} \times \Psi N \cup N \Psi \cup \{\langle \epsilon, \epsilon \rangle\}$$

Where  $\square$  represents identity permutation and  $\langle \rangle$  represents inversion permutation,  $\Psi = \Sigma \cup \{\epsilon\} \times \Delta \cup \{\epsilon\}$  is a possibly empty biterminal, and  $\epsilon$  is the empty string.

Graphically, a rule will be represented as an ITG rule:

$$\begin{aligned} \langle A, \square, B \langle e, f \rangle \rangle &= A \rightarrow [ B e/f ] \\ \langle A, \langle \rangle, \langle e, f \rangle B \rangle &= A \rightarrow \langle e/f B \rangle \\ \langle A, \square, \langle \epsilon, \epsilon \rangle \rangle &= A \rightarrow \epsilon/\epsilon \end{aligned}$$

As with ITGs, productions with only biterminals will be represented without their permutation, as any such rule can be trivially rewritten into inverted or identity form.

**Definition 4.** An  $\epsilon$ -free LITG is an LITG where no rule may rewrite one nonterminal into another nonterminal only. Formally, the rule set is constrained so that

$$R \cap N \times \{\langle \rangle, \langle \rangle\} \times (\{\langle \epsilon, \epsilon \rangle\} B \cup B \{\langle \epsilon, \epsilon \rangle\}) = \emptyset$$

The LITG presented in Saers et al. (2010) is thus an  $\epsilon$ -free LITG in normal form, since it has the following thirteen rule forms (of which 8 are meaningful, 1 is only used to terminate generation and 4 are redundant):

$$\begin{array}{l} A \rightarrow [ e/f B ] \\ A \rightarrow \langle e/f B \rangle \\ A \rightarrow [ B e/f ] \\ A \rightarrow \langle B e/f \rangle \\ A \rightarrow [ e/\epsilon B ] \quad | \quad A \rightarrow \langle e/\epsilon B \rangle \\ A \rightarrow [ B e/\epsilon ] \quad | \quad A \rightarrow \langle B e/\epsilon \rangle \\ A \rightarrow [ \epsilon/f B ] \quad | \quad A \rightarrow \langle \epsilon/f B \rangle \\ A \rightarrow [ B \epsilon/f ] \quad | \quad A \rightarrow \langle B \epsilon/f \rangle \\ A \rightarrow \epsilon/\epsilon \end{array}$$

All the singleton rules can be expressed either in straight or inverted form, but the result of applying the two rules are the same.

**Lemma 1.** Any LITG in normal form can be expressed as an LTG in normal form.

*Proof.* The above LITG can be rewritten in LTG form as follows:

$$\begin{array}{l} A \rightarrow [ e/f B ] = A \rightarrow e/f B \\ A \rightarrow \langle e/f B \rangle = A \rightarrow e/\epsilon B \epsilon/f \\ A \rightarrow [ B e/f ] = A \rightarrow B e/f \\ A \rightarrow \langle B e/f \rangle = A \rightarrow \epsilon/f B e/\epsilon \\ A \rightarrow [ e/\epsilon B ] = A \rightarrow e/\epsilon B \\ A \rightarrow [ B e/\epsilon ] = A \rightarrow B e/\epsilon \\ A \rightarrow [ \epsilon/f B ] = A \rightarrow \epsilon/f B \\ A \rightarrow [ B \epsilon/f ] = A \rightarrow B \epsilon/f \\ A \rightarrow \epsilon/\epsilon = A \rightarrow \epsilon/\epsilon \end{array}$$

To account for all LITGs in normal form, the following two non- $\epsilon$ -free rules also needs to be accounted for:

$$\begin{array}{l} A \rightarrow [ B ] = A \rightarrow B \\ A \rightarrow \langle B \rangle = A \rightarrow B \end{array}$$

□

**Lemma 2.** Any LTG in normal form can be expressed as an LITG in normal form.

*Proof.* An LTG in normal form has two rules, which can be rewritten in LITG form, either as straight or inverted rules as follows

$$\begin{array}{l} A \rightarrow x/p B y/q = A \rightarrow [ x/p \bar{B} ] \\ \bar{B} \rightarrow [ B y/q ] \\ = A \rightarrow \langle x/q \bar{B} \rangle \\ \bar{B} \rightarrow \langle B y/p \rangle \\ A \rightarrow \epsilon/\epsilon = A \rightarrow \epsilon/\epsilon \end{array}$$

□

**Theorem 1.** LTGs in normal form and LITGs in normal form express the same class of transductions.

*Proof.* Follows from lemmas 1 and 2. □

By theorem 1 everything concerning LTGs is also applicable to LITGs, and an LTG can be expressed in LITG form when convenient, and vice versa.

## 4 Pruning the Alignment Space

The alignment space for a transduction grammar is the combinations of the parse spaces of the sentence pair. Let  $e$  be the  $E$  sentence, and  $f$  be the  $F$  sentence. The parse spaces would be  $\mathcal{O}(|e|^2)$  and  $\mathcal{O}(|f|^2)$  respectively, and the combination of these spaces would be  $\mathcal{O}(|e|^2 \times |f|^2)$ , or  $\mathcal{O}(n^4)$  if we assume  $n$  to be proportional to the sentence lengths. In the case of LTGs, this space is searched linearly, giving time complexity  $\mathcal{O}(n^4)$ , and in the case of ITGs there is branching within both parse spaces, adding an order of magnitude each, giving a total time complexity of  $\mathcal{O}(n^6)$ . There is, in other words, a tight connection between the alignment space and the time complexity of the biparsing algorithm. Furthermore, most of this alignment space is clearly useless. Consider the case where the entire  $F$  sentence is deleted, and the entire  $E$  sentence is simply inserted. Although it is possible that it is allowed by the grammar, it should have a negligible probability (since it is clearly a translation strategy that generalize poorly), and could, for all practical reasons, be ignored.

Language pair	Bisentences	Tokens
Spanish–English	108,073	1,466,132
French–English	95,990	1,340,718
German–English	115,323	1,602,781

Table 1: Size of training data.

Saers et al. (2009) present a scheme for pruning away most of the points in the alignment space. Parse items are binned according to coverage (the total number of words covered), and each bin is restricted to carry a maximum of  $b$  items. Any items that do not fit in the bins are excluded from further analysis. To decide which items to keep, inside probability is used. This pruning scheme effectively linearizes the alignment space, as it will be of size  $\mathcal{O}(nb)$ , regardless of what type grammar is used. An ITG can thus be biparsed in cubic time, and an LTG in linear time.

## 5 Seeding an ITG with an LTG

Since LTGs are a subclass of ITGs, it would be possible to convert an LTG to a ITG. This could save a lot of time, since LTGs are *much* faster to induce from corpora than ITGs.

Converting a BLTG to a BITG is fairly straight forward. Consider the BLTG rule

$$X \rightarrow [ e/f X ]$$

To convert it to BITG in two-normal form, the biterminal has to be factored out. Replacing the biterminal with a temporary symbol  $\bar{X}$ , and introducing a rule that rewrites this temporary symbol to the replaced biterminal produces two rules:

$$\begin{aligned} X &\rightarrow [ \bar{X} X ] \\ \bar{X} &\rightarrow e/f \end{aligned}$$

This is no longer a bracketing grammar since there are two nonterminals, but equating  $\bar{X}$  to  $X$  restores this property. An analogous procedure can be applied in the case where the nonterminal comes before the biterminal, as well as for the inverting cases.

When converting stochastic LTGs, the probability mass of the SLTG rule has to be distributed

to two SITG rules. The fact that the LTG rule  $X \rightarrow \epsilon/\epsilon$  lacks correspondence in ITGs has to be weighted in as well. In this paper we took the maximum entropy approach and distributed the probability mass uniformly. This means defining the probability mass function  $p'$  for the new SBITG from the probability mass function  $p$  of the original SBLTG such that:

$$\begin{aligned} p'(X \rightarrow [ X X ]) &= \sum_{e/f} \left[ \frac{\sqrt{\frac{p(X \rightarrow [ e/f X ]) }{1-p(X \rightarrow \epsilon/\epsilon)}}}{\sqrt{\frac{p(X \rightarrow [ X e/f ]) }{1-p(X \rightarrow \epsilon/\epsilon)}}} \right] \\ p'(X \rightarrow \langle X X \rangle) &= \sum_{e/f} \left[ \frac{\sqrt{\frac{p(X \rightarrow \langle e/f X \rangle) }{1-p(X \rightarrow \epsilon/\epsilon)}}}{\sqrt{\frac{p(X \rightarrow \langle X e/f \rangle) }{1-p(X \rightarrow \epsilon/\epsilon)}}} \right] \\ p'(X \rightarrow e/f) &= \frac{\sqrt{\frac{p(X \rightarrow [ e/f X ]) }{1-p(X \rightarrow \epsilon/\epsilon)}}}{\sqrt{\frac{p(X \rightarrow [ X e/f ]) }{1-p(X \rightarrow \epsilon/\epsilon)}} + \sqrt{\frac{p(X \rightarrow \langle e/f X \rangle) }{1-p(X \rightarrow \epsilon/\epsilon)}} + \sqrt{\frac{p(X \rightarrow \langle X e/f \rangle) }{1-p(X \rightarrow \epsilon/\epsilon)}}} \end{aligned}$$

## 6 Setup

The aim of this paper is to compare the alignments from SBITG and SBLTG to those from GIZA++, and to study the impact of pruning on efficiency and translation quality. Initial grammars will be estimated by counting co-occurrences in the training corpus, after which expectation-maximization (EM) will be used to refine the initial estimate. At the last iteration, the one-best parse of each sentence will be considered as the word alignment of that sentence.

In order to keep the experiments comparable, relatively small corpora will be used. If larger corpora were used, it would not be possible to get any results for unpruned SBITGs because of the prohibitive time complexity. The Europarl corpus (Koehn, 2005) was used as a starting point, and then all sentence pairs where one of the sentences were longer than 10 tokens were filtered

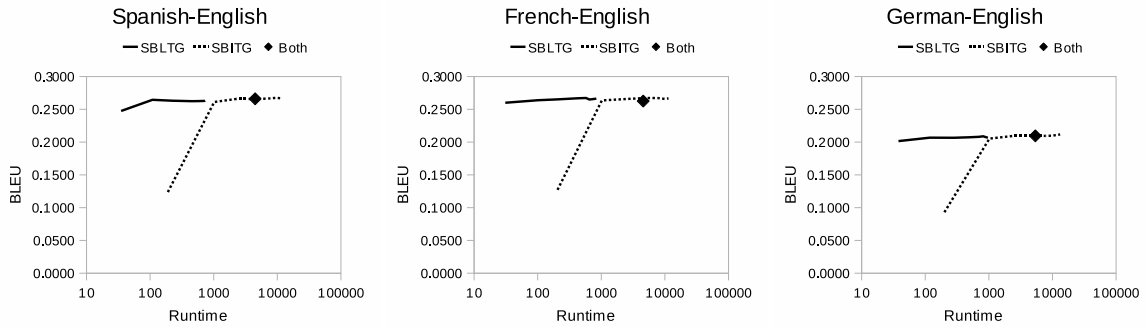


Figure 1: Trade-offs between translation quality (as measured by BLEU) and biparsing time (in seconds plotted on a logarithmic scale) for SBLTGs, SBITGs and the combination.

System	Beam size						
	1	10	25	50	75	100	$\infty$
BLEU							
SBITG	0.1234	0.2608	<b>0.2655</b>	<b>0.2653</b>	<b>0.2661</b>	<b>0.2671</b>	<b>0.2663</b>
SBLTG	0.2574	<b>0.2645</b>	0.2631	0.2624	0.2625	0.2633	0.2628
GIZA++	<b>0.2597</b>	0.2597	0.2597	0.2597	0.2597	0.2597	0.2597
NIST							
SBITG	3.9705	6.6439	<b>6.7312</b>	<b>6.7101</b>	<b>6.7329</b>	<b>6.7445</b>	<b>6.6793</b>
SBLTG	6.6023	<b>6.6800</b>	6.6657	6.6637	6.6714	6.6863	6.6765
GIZA++	<b>6.6464</b>	6.6464	6.6464	6.6464	6.6464	6.6464	6.6464
Training times							
SBITG	03:10	17:00	38:00	1:20:00	2:00:00	2:40:00	3:20:00
SBLTG	35	1:49	3:40	7:33	9:44	12:13	11:59

Table 2: Results for the Spanish–English translation task.

out (see table 1). The GIZA++ system was built according to the instructions for creating a baseline system for the Fifth Workshop on Statistical Machine Translation (WMT’10),<sup>1</sup> but the above corpora were used instead of those supplied by the workshop. This includes word alignment with GIZA++, a 5-gram language model built with SRILM (Stolcke, 2002) and parameter tuning with MERT (Och, 2003). To carry out the actual translations, Moses (Koehn et al., 2007) was used. The SBITG and SBLTG systems were built in exactly the same way, except that the alignments from GIZA++ were replaced by those from the respective grammars.

In addition to trying out exhaustive biparsing

for SBITGs and SBLTGs on three different translation tasks, several different levels of pruning were tried (1, 10, 25, 50, 75 and 100). We also used the grammar induced from SBLTGs with a beam size of 25 to seed SBITGs (see section 5), which were then run for an additional iteration of EM, also with beam size 25.

All systems are evaluated with BLEU (Papineni et al., 2002) and NIST (Doddington, 2002).

## 7 Results

The results for the three different translation tasks are presented in Tables 2, 3 and 4. It is interesting to note that the trend they portray is quite similar. When the beam is very narrow, GIZA++ is better, but already at beam size 10, both transduction grammars are superior. Con-

<sup>1</sup><http://www.statmt.org/wmt10/>

System	Beam size						
	1	10	25	50	75	100	$\infty$
BLEU							
SBITG	0.1268	0.2632	<b>0.2654</b>	<b>0.2669</b>	0.2668	0.2655	<b>0.2663</b>
SBLTG	0.2600	<b>0.2638</b>	0.2651	0.2668	<b>0.2672</b>	<b>0.2662</b>	0.2649
GIZA++	<b>0.2603</b>	0.2603	0.2603	0.2603	0.2603	0.2603	0.2603
NIST							
SBITG	4.0849	6.7136	<b>6.7913</b>	<b>6.8065</b>	<b>6.8068</b>	<b>6.8088</b>	<b>6.8151</b>
SBLTG	6.6814	<b>6.7608</b>	6.7656	6.7992	6.8020	6.7925	6.7784
GIZA++	<b>6.6907</b>	6.6907	6.6907	6.6907	6.6907	6.6907	6.6907
Training times							
SBITG	03:25	17:00	42:00	1:25:00	2:10:00	2:45:00	3:10:00
SBLTG	31	1:41	3:25	7:06	9:35	13:56	10:52

Table 3: Results for the French–English translation task.

System	Beam size						
	1	10	25	50	75	100	$\infty$
BLEU							
SBITG	0.0926	0.2050	<b>0.2091</b>	<b>0.2090</b>	<b>0.2091</b>	<b>0.2094</b>	<b>0.2113</b>
SBLTG	0.2015	<b>0.2067</b>	0.2066	0.2073	0.2080	0.2066	0.2088
GIZA++	<b>0.2059</b>	0.2059	0.2059	0.2059	0.2059	0.2059	0.2059
NIST							
SBITG	3.4297	5.8743	<b>5.9292</b>	5.8947	5.8955	<b>5.9086</b>	<b>5.9380</b>
SBLTG	5.7799	<b>5.8819</b>	5.8882	<b>5.8963</b>	<b>5.9252</b>	5.8757	5.9311
GIZA++	<b>5.8668</b>	5.8668	5.8668	5.8668	5.8668	5.8668	5.8668
Training times							
SBITG	03:20	17:00	41:00	1:25:00	2:10:00	2:45:00	3:40:00
SBLTG	38	1:58	4:52	8:08	11:42	16:05	13:32

Table 4: Results for the German–English translation task.

sistent with Saers et al. (2009), SBITG has a sharp rise in quality going from beam size 1 to 10, and then a gentle slope up to beam size 25, after which it levels out. SBLTG, on the other hand start out at a respectable level, and goes up a gentle slope from beam size 1 to 10, after which is level out. This is an interesting observation, as it suggests that SBLTG reaches its optimum with a lower beam size (although that optimum is lower than that of SBITG). The trade-off between quality and time can now be extended beyond beam size to include grammar choice. In Figure 1, run times are plotted against BLEU scores to illus-

trate this trade-off. It is clear that SBLTGs are indeed much faster than SBITGs, the only exception is when SBITGs are run with  $b = 1$ , but then the BLEU score is so low that is is not worth considering.

The time may seem inconsistent between  $b = 100$  and  $b = \infty$  for SBLTG, but the extra time for the tighter beam is because of beam management, which the exhaustive search doesn’t bother with.

In table 5 we compare the pure approaches to one where an LTG was trained during 10 iterations of EM and then used to seed (see sec-

Translation task	System	BLEU	NIST	Total time
Spanish–English	SBLTG	0.2631	6.6657	36:40
	SBITG	0.2655	<b>6.7312</b>	6:20:00
	Both	<b>0.2660</b>	6.7124	1:14:40
French–English	SBLTG	0.2651	6.7656	34:10
	SBITG	<b>0.2654</b>	<b>6.7913</b>	7:00:00
	Both	0.2625	6.7609	1:16:10
German–English	SBLTG	0.2066	5.8882	48:52
	SBITG	0.2091	<b>5.9292</b>	6:50:00
	Both	<b>0.2095</b>	5.9224	1:29:40

Table 5: Results for seeding an SBITG with an SBLTG (Both) compared to the pure approach. Total time refers to 10 iterations of EM training for SBITG and SBLTG respectively, and 10 iterations of SBLTG and one iteration of SBITG training for the combined system.

tion 5) an SBITG, which was then trained for one iteration of EM. Although the differences are fairly small, German–English and Spanish–English seem to reach the level of SBITG, whereas French–English is actually hurt. The big difference is in time, since the combined system needs about a fifth of the time the SBITG-based system needs. This phenomenon needs to be more thoroughly examined.

It is also worth noting that GIZA++ was beaten by an aligner that used less than 20 minutes (less than 2 minutes per iteration and at most 10 iterations) to align the corpus.

## 8 Conclusions

In this paper we have introduced the bilingual version of linear grammar: Linear Transduction Grammars, and found that they generate the same class of transductions as Linear Inversion Transduction Grammars. We have also compared Stochastic Bracketing versions of ITGs and LTGs to GIZA++ on three word alignment tasks. The efficiency issues with transduction grammars have been addressed by pruning, and the conclusion is that there is a trade-off between run time and translation quality. A part of the trade-off is choosing which grammar framework to use, as LTGs are faster but not as good as ITGs. It also seems possible to take a short-cut in this trade-off by starting out with an LTG and converting it to an ITG. We have also showed that it is

possible to beat the translation quality of GIZA++ with a quite fast transduction grammar.

## Acknowledgments

This work was funded by the Swedish National Graduate School of Language Technology (GSLT), the Defense Advanced Research Projects Agency (DARPA) under GALE Contracts No. HR0011-06-C-0022 and No. HR0011-06-C-0023, and the Hong Kong Research Grants Council (RGC) under research grants GRF621008, DAG03/04.EG09, RGC6256/00E, and RGC6083/99E. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency. The computations were performed on UPPMAX resources under project p2007020.

## References

- Aho, Alfred V. Ullman, Jeffrey D. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice-Hall, Inc., Upper Saddle River, NJ.
- Brown, Peter F., 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.
- Cocke, John. 1969. *Programming languages and their compilers: Preliminary notes*. Courant Insti-



- tute of Mathematical Sciences, New York University.
- Dodgington, George. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of Human Language Technology conference (HLT-2002)*, San Diego, California.
- Earley, Jay. 1970. An efficient context-free parsing algorithm. *Communications of the Association for Computer Machinery*, 13(2):94–102.
- Haghighi, Aria, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised ITG models. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 923–931, Suntec, Singapore, August.
- Huang, Liang, Hao Zhang, Daniel Gildea, and Kevin Knight. 2009. Binarization of synchronous context-free grammars. *Computational Linguistics*, 35(4):559–595.
- Kasami, Tadao. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-00143, Air Force Cambridge Research Laboratory.
- Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June.
- Koehn, Philipp. 2005. Europarl: A parallel corpus for statistical machine translation. In *Machine Translation Summit X*, Phuket, Thailand, September.
- Lewis, Philip M. and Richard E. Stearns. 1968. Syntax-directed transduction. *Journal of the Association for Computing Machinery*, 15(3):465–488.
- Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.
- Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July.
- Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, July.
- Saers, Markus and Dekai Wu. 2009. Improving phrase-based translation via word alignments from Stochastic Inversion Transduction Grammars. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation (SSST-3) at NAACL HLT 2009*, pages 28–36, Boulder, Colorado, June.
- Saers, Markus, Joakim Nivre, and Dekai Wu. 2009. Learning Stochastic Bracketing Inversion Transduction Grammars with a cubic time biparsing algorithm. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 29–32, Paris, France, October.
- Saers, Markus, Joakim Nivre, and Dekai Wu. 2010. Word alignment with Stochastic Bracketing Linear Inversion Transduction Grammar. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Los Angeles, California, June.
- Stolcke, Andreas. 2002. SRILM – an extensible language modeling toolkit. In *International Conference on Spoken Language Processing*, Denver, Colorado, September.
- Vogel, Stephan, Hermann Ney, and Christoph Tillmann. 1996. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics*, pages 836–841, Morristown, New Jersey.
- Wu, Dekai. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.
- Younger, Daniel H. 1967. Recognition and parsing of context-free languages in time  $n^3$ . *Information and Control*, 10(2):189–208.
- Zhang, Hao, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *Proceedings of ACL-08: HLT*, pages 97–105, Columbus, Ohio, June.