

# Dependency Treelet Translation: The convergence of statistical and example-based machine-translation?

Arul Menezes and Chris Quirk

Microsoft Research

One Microsoft Way, Redmond, WA 98052

{arulm,chrisk}@microsoft.com

## Abstract

We describe a novel approach to machine translation that combines the strengths of the two leading corpus-based approaches: Phrasal SMT and EBMT. We use a syntactically informed decoder and reordering model based on the source dependency tree, in combination with conventional SMT models to incorporate the power of phrasal SMT with the linguistic generality available in a parser. We show that this approach significantly outperforms a leading string-based Phrasal SMT decoder and an EBMT system. We present results from two radically different language pairs, and investigate the sensitivity of this approach to parse quality by using two distinct parsers and oracle experiments. We also validate our automated BLEU scores with a small human evaluation.

## 1. Introduction

Current example-based (EBMT) and statistical (SMT) machine translation systems both use phrases learned from parallel corpora, yet while the two approaches are closer than ever, some critical differences remain. (Way & Gough, 2005) On the one hand, while statistical systems excel at producing correct, even idiomatic translations at the local level, they are still challenged by many linguistic phenomena, such as global constituent ordering. While SMT excels at translating domain-specific terminology and fixed phrases, grammatical generalizations are poorly captured and often mangled in translation (Thurmair, 04).

On the other hand, many EBMT systems do not fully exploit the power that results from a combination of multiple powerful statistical models. In particular, we believe that the recent dominance of SMT systems in competitive

evaluations indicates that an end-to-end search over a weighted linear combination of statistical models is essential for high-quality translation. However, there is no indication that these models must necessarily be linguistically uninformed.

### 1.1. Limitations of string-based phrasal SMT

State-of-the-art phrasal SMT systems such as (Koehn et al., 03) and (Vogel et al., 03) model translations of *phrases* (here, strings of adjacent words, not syntactic constituents) rather than individual words. Arbitrary reordering of words is allowed within memorized phrases, but typically only a small amount of phrase reordering is allowed, modeled in terms of offset positions at the string level. This reordering model is very limited in terms of linguistic generalizations. For instance, when translating English to Japanese, an ideal system would automatically learn large-scale typological differences: English SVO clauses generally become Japanese SOV clauses, English post-modifying prepositional phrases become Japanese pre-modifying postpositional phrases, etc. A phrasal SMT system may learn the internal reordering of specific common phrases, but it cannot generalize to unseen phrases that share the same linguistic structure.

In addition, these systems are limited to phrases contiguous in both source and target, and thus cannot learn the generalization that English *not* may translate as French *ne...pas* except in the context of specific intervening words.

### 1.2. Previous work on syntactic SMT and statistical EBMT

The hope in the SMT community has been that the incorporation of syntax would address these issues, but that promise has yet to be realized<sup>1</sup>.

---

<sup>1</sup> Note that as the focus of this paper is decoding, we do not discuss the large body of work incorporating syntax into the word alignment process.

One simple means of incorporating syntax into SMT decoding is by re-ranking the  $n$ -best list of a baseline SMT system using various syntactic models, but Och et al. (04) found very little positive impact with this approach. However, an  $n$ -best list of even 16,000 translations captures only a tiny fraction of the ordering possibilities of a 20 word sentence; re-ranking provides the syntactic model no opportunity to boost or prune large sections of that search space.

Inversion Transduction Grammars (Wu, 97), or ITGs, treat translation as a process of parallel parsing of the source and target language via a synchronized grammar. To make this process computationally efficient, however, some severe simplifying assumptions are made, such as using a single non-terminal label. This results in the model simply learning a very high level preference regarding how often nodes should switch order without any contextual information. Also these translation models are intrinsically word-based; phrasal combinations are not modeled directly, and results have not been competitive with the top phrasal SMT systems.

Along similar lines, Alshawi et al. (2000) treat translation as a process of simultaneous induction of source and target dependency trees using head-transduction; again, no separate parser is used.

Yamada and Knight (01) employ a parser in the target language to train probabilities on a set of operations that convert a target language tree to a source language string. This improves fluency slightly (Charniak et al., 03), but fails to significantly impact overall translation quality. This may be because the parser is applied to MT output, which is notoriously unlike native language, and no additional insight is gained via source language analysis.

Lin (04) translates dependency trees using paths. This is the first attempt to incorporate large phrasal SMT-style memorized patterns together with a separate source dependency parser and SMT models. However the phrases are limited to linear paths in the tree, the only SMT model used is a maximum likelihood channel model and there is no ordering model. Reported BLEU scores are far below the leading phrasal SMT systems.

Aue et al. (04) recently reported incorporating a logical form (LF) or dependency tree-based statistical language model into an existing EBMT system. MSR-MT (Menezes & Richardson, 03)

parses both source and target languages to obtain a logical form (LF), and translates source LFs using memorized aligned LF examples to produce a target LF. It utilizes a separate sentence realization component (Ringger et al., 04) to turn this into a target sentence. As a result, Aue could not use an end-to-end search over a linear combination of models, and the simple addition of a single target language model did not provide much improvement.

## 2. Dependency Treelet Translation

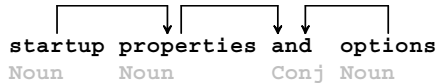
In this paper we propose a novel dependency tree-based approach to phrasal SMT which uses tree-based ‘phrases’ and a tree-based ordering model in combination with conventional SMT models to produce translations significantly better than a leading string-based system.

Our system employs a source-language dependency parser, a target language word segmentation component, and an unsupervised word alignment component to learn treelet translations from a parallel sentence-aligned corpus. We begin by parsing the source text to obtain dependency trees and word-segmenting the target side, then applying an off-the-shelf word alignment component to the bitext.

The word alignments are used to project the source dependency parses onto the target sentences. From this aligned parallel dependency corpus we extract a treelet translation model incorporating source and target treelet pairs, where a *treelet* is defined to be an arbitrary connected subgraph of the dependency tree. A unique feature is that we allow treelets with a wildcard root, effectively allowing mappings for siblings in the dependency tree. This allows us to model important phenomena, such as *not ... → ne...pas*. We also train a variety of statistical models on this aligned dependency tree corpus, including a channel model and an order model.

To translate an input sentence, we parse the sentence, producing a dependency tree for that sentence. We then employ a decoder to find a combination and ordering of treelet translation pairs that cover the source tree and are optimal according to a set of models that are combined in a log-linear framework as in (Och, 03).

This approach offers the following advantages over string-based SMT systems: Instead of



**Figure 1.** An example dependency tree.

limiting learned phrases to contiguous word sequences, we allow translation by all possible phrases that form connected subgraphs (treelets) in the source and target dependency trees. This is a powerful extension: the vast majority of surface-contiguous phrases are also treelets of the tree; in addition, we gain discontinuous phrases, including combinations such as verb-object, article-noun, adjective-noun etc. regardless of the number of intervening words.

Another major advantage is the ability to employ more powerful models for reordering source language constituents. These models can incorporate information from the source analysis. For example, we may model directly the probability that the translation of an object of a preposition in English should precede the corresponding postposition in Japanese, or the probability that a pre-modifying adjective in English translates into a post-modifier in French.

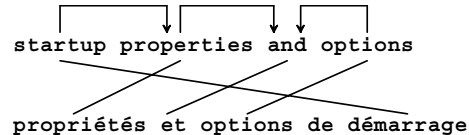
## 2.1. Parsing and alignment

We require a source language dependency parser that produces unlabeled, ordered dependency trees and annotates each source word with a part-of-speech (POS). An example dependency tree is shown in Figure 1. The arrows indicate the head annotation, and the POS for each candidate is listed underneath. For the target language we only require word segmentation.

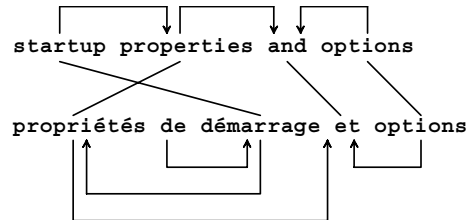
To obtain word alignments we currently use GIZA++ (Och & Ney, 03). We follow the common practice of deriving many-to-many alignments by running the IBM models in both directions and combining the results heuristically. Our heuristics differ in that they constrain many-to-one alignments to be contiguous in the source dependency tree. A detailed description of these heuristics can be found in (Quirk et al, 2004).

## 2.2. Projecting dependency trees

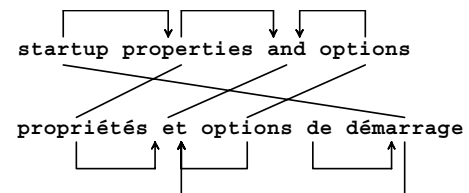
Given a word aligned sentence pair and a source dependency tree, we use the alignment to project the source structure onto the target sentence. One-to-one alignments project directly to create a



(a) Word alignment.



(b) Dependencies after initial projection.



(c) Dependencies after reattachment step.

**Figure 2.** Projection of dependencies.

target tree isomorphic to the source. Many-to-one alignments project similarly; since the ‘many’ source nodes are connected in the tree, they act as if condensed into a single node. In the case of one-to-many alignments we project the source node to the rightmost<sup>2</sup> of the ‘many’ target words, and make the rest of the target words dependent on it.

Unaligned target words<sup>3</sup> are attached into the dependency structure as follows: assume there is an unaligned word  $t_j$  in position  $j$ . Let  $i < j$  and  $k > j$  be the target positions closest to  $j$  such that  $t_i$  depends on  $t_k$  or vice versa: attach  $t_j$  to the lower of  $t_i$  or  $t_k$ . If all the nodes to the left (or right) of position  $j$  are unaligned, attach  $t_j$  to the left-most (or right-most) word that is aligned.

The target dependency tree created in this process may not read off in the same order as the target string, since our alignments do not enforce phrasal cohesion. For instance, consider the projection of the parse in Figure 1 using the word alignment in Figure 2a. Our algorithm produces the dependency tree in Figure 2b. If we read off the leaves in a left-to-right in-order traversal, we

<sup>2</sup> If the target language is Japanese, leftmost may be more appropriate.

<sup>3</sup> Source unaligned nodes do not present a problem, with the exception that if the root is unaligned, the projection process produces a forest of target trees anchored by a dummy root.

do not get the original input string: *de démarrage* appears in the wrong place.

A second reattachment pass corrects this situation. For each node in the wrong order, we reattach it to the lowest of its ancestors such that it is in the correct place relative to its siblings and parent. In Figure 2c, reattaching *démarrage* to *et* suffices to produce the correct order.

### 2.3. Extracting treelet translation pairs

From the aligned pairs of dependency trees we extract all pairs of aligned source and target treelets along with word-level alignment linkages, up to a configurable maximum size. We also keep treelet counts for maximum likelihood estimation.

### 2.4. Order model

Phrasal SMT systems often use a model to score the ordering of a set of phrases. One approach is to penalize any deviation from monotone decoding; another is to estimate the probability that a source phrase in position  $i$  translates to a target phrase in position  $j$  (Koehn et al., 03).

We attempt to improve on these approaches by incorporating syntactic information. Our model assigns a probability to the order of a target tree given a source tree. Under the assumption that constituents generally move as a whole, we predict the probability of each given ordering of modifiers independently. That is, we make the following simplifying assumption (where  $c$  is a function returning the set of nodes modifying  $t$ ):

$$P(\text{order}(T) | S, T) = \prod_{t \in T} P(\text{order}(c(t)) | S, T)$$

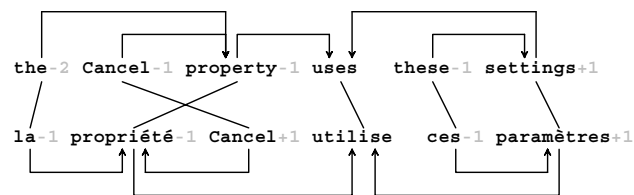
Furthermore, we assume that the position of each child can be modeled independently in terms of a head-relative position:

$$P(\text{order}(c(t)) | S, T) = \prod_{m \in c(t)} P(\text{pos}(m, t) | S, T)$$

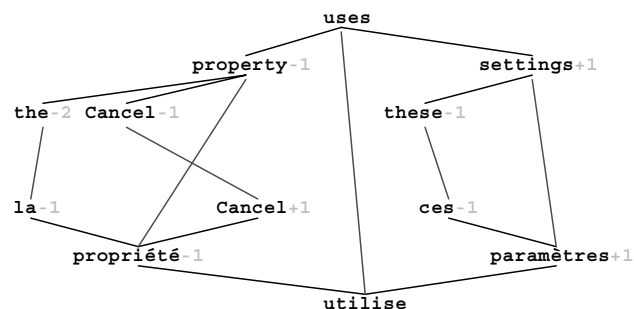
Figure 3a demonstrates an aligned dependency tree pair annotated with head-relative positions; Figure 3b presents the same information in an alternate tree-like representation.

We currently use a small set of features reflecting very local information in the dependency tree to model  $P(\text{pos}(m, t) | S, T)$ :

- The lexical items of the head and modifier.
- The lexical items of the source nodes aligned to the head and modifier.



(a) Head annotation representation



(b) Branching structure representation.

**Figure 3.** Aligned dependency tree pair, annotated with head-relative positions

- The part-of-speech ("cat") of the source nodes aligned to the head and modifier.
- The head-relative position of the source node aligned to the source modifier. (One can also include features of siblings to produce a Markov ordering model. However, we found that this had little impact in practice).

As an example, consider the children of *propriété* in Figure 3. The head-relative positions of its modifiers *la* and *Cancel* are -1 and +1, respectively. Thus we try to predict as follows:

$$P(\text{pos}(m_1) = -1 | \text{lex}(m_1) = "la", \text{lex}(h) = "propriété", \text{lex}(\text{src}(m_1)) = "the", \text{lex}(\text{src}(h)) = "property", \text{cat}(\text{src}(m_1)) = \text{Determiner}, \text{cat}(\text{src}(h)) = \text{Noun}, \text{position}(\text{src}(m_1)) = -2) \cdot$$

$$P(\text{pos}(m_2) = +1 | \text{lex}(m_2) = "Cancel", \text{lex}(h) = "propriété", \text{lex}(\text{src}(m_2)) = "Cancel", \text{lex}(\text{src}(h)) = "property", \text{cat}(\text{src}(m_2)) = \text{Noun}, \text{cat}(\text{src}(h)) = \text{Noun}, \text{position}(\text{src}(m_2)) = -1)$$

The training corpus acts as a supervised training set: we extract a training feature vector from each of the target language nodes in the aligned dependency tree pairs. Together these feature vectors are used to train a decision tree (Chickering, 02). The distribution at each leaf of the DT can be used to assign a probability to each possible target language position. A more detailed description is available in (Quirk et al, 2004).

## 2.5. Other models

*Channel Models:* We incorporate two distinct channel models, a maximum likelihood estimate (MLE) model and a model computed using Model-1 word-to-word alignment probabilities as in (Vogel et al., 03). The MLE model effectively captures non-literal phrasal translations such as idioms, but suffers from data sparsity. The word-to-word model does not typically suffer from data sparsity, but prefers more literal translations.

Given a set of treelet translation pairs that cover a given input dependency tree and produce a target dependency tree, we model the probability of source given target as the product of the individual treelet translation probabilities: we assume a uniform probability distribution over the decompositions of a tree into treelets.

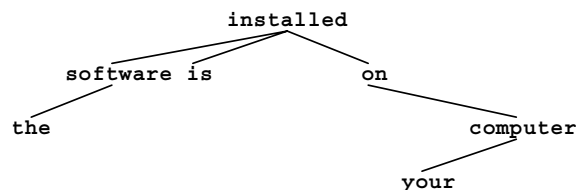
*Target Model:* Given an ordered target language dependency tree, it is trivial to read off the surface string. We evaluate this string using a trigram model with modified Kneser-Ney smoothing.

*Miscellaneous Feature Functions:* The log-linear framework allows us to incorporate other feature functions as ‘models’ in the translation process. For instance, using fewer, larger treelet translation pairs often provides better translations, since they capture more context and allow fewer possibilities for search and model error. Therefore we add a feature function that counts the number of phrases used. We also add a feature that counts the number of target words; this acts as an insertion/deletion bonus/penalty.

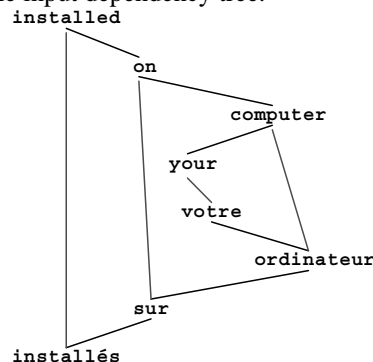
## 3. Decoding

The challenge of tree-based decoding is that the traditional left-to-right decoding approach of string-based systems is inapplicable. Additional challenges are posed by the need to handle treelets—perhaps discontinuous or overlapping—and a combinatorially explosive ordering space.

Our decoding approach is influenced by ITG (Wu, 97) with several important extensions. First, we employ treelet translation pairs instead of single word translations. Second, instead of modeling rearrangements as either preserving source order or swapping source order, we allow the dependents of a node to be ordered in any arbitrary manner and use the order model described in section 2.4 to estimate probabilities.



(a) Example input dependency tree.



(b) Example treelet translation pair.

**Figure 4.** Example decoder structures.

Finally, we use a log-linear framework for model combination that allows any amount of other information to be modeled.

We will initially approach the decoding problem as a bottom up, exhaustive search. We define the set of all possible treelet translation pairs of the subtree rooted at each input node in the following manner: A treelet translation pair  $x$  is said to *match* the input dependency tree  $S$  iff there is some connected subgraph  $S'$  that is identical to the source side of  $x$ . We say that  $x$  *covers* all the nodes in  $S'$  and is *rooted* at source node  $s$ , where  $s$  is the root of matched subgraph  $S'$ .

We first find all treelet translation pairs that match the input dependency tree. Each matched pair is placed on a list associated with the input node where the match is rooted. Moving bottom-up through the input dependency tree, we compute a list of *candidate translations* for the input subtree rooted at each node  $s$ , as follows:

Consider in turn each treelet translation pair  $x$  rooted at  $s$ . The treelet pair  $x$  may cover only a portion of the input subtree rooted at  $s$ . Find all descendants  $s'$  of  $s$  that are *not* covered by  $x$ , but whose parent  $s''$  is covered by  $x$ . At each such node  $s''$  look at all interleavings of the children of  $s''$  specified by  $x$ , if any, with each translation  $t'$  from the candidate translation list<sup>4</sup> of each child

<sup>4</sup> Computed by the previous application of this procedure to  $s'$  during the bottom-up traversal.

$s'$ . Each such interleaving is scored using the models previously described and added to the candidate translation list for that input node. The resultant translation is the best scoring candidate for the root input node.

As an example, see the example dependency tree in Figure 4a and treelet translation pair in 4b. This treelet translation pair covers all the nodes in 4a except the subtrees rooted at *software* and *is*. We first compute (and cache) the candidate translation lists for the subtrees rooted at *software* and *is*, then construct full translation candidates by attaching those subtree translations to *installés* in all possible ways. The order of *sur* relative to *installés* is fixed; it remains to place the translated subtrees for *the software* and *is*. Note that if  $c$  is the count of children specified in the mapping and  $r$  is the count of subtrees translated via recursive calls, then there are  $(c+r+1)!/(c+1)!$  orderings. Thus  $(1+2+1)!/(1+1)! = 12$  candidate translations are produced for each combination of translations of *the software* and *is*.

### 3.1. Optimality-preserving optimizations

#### Dynamic Programming

Converting this exhaustive search to dynamic programming relies on the observation that scoring a translation candidate at a node depends on the following information from its descendants: the order model requires features from the root of a translated subtree, and the target language model is affected by the first and last two words in each subtree. Therefore, we need to keep the best scoring translation candidate for a given subtree for each combination of (head, leading bigram, trailing bigram), which is, in the worst case,  $O(V^5)$ , where  $V$  is the vocabulary size. The dynamic programming approach therefore does not allow for great savings in practice because a trigram target language model forces consideration of context external to each subtree.

#### 3.2. Lossy optimizations

The following optimizations do not preserve optimality, but work well in practice.

#### *N*-best lists

Instead of keeping the full list of translation candidates for a given input node, we keep a top-scoring subset of the candidates. While the

decoder is no longer guaranteed to find the optimal translation, in practice the quality impact is minimal with a list size  $\geq 10$  (see Table 5.6).

*Variable-sized n-best lists:* A further speedup can be obtained by noting that the number of translations using a given treelet pair is exponential in the number of subtrees of the input not covered by that pair. To limit this explosion we vary the size of the  $n$ -best list on any recursive call in inverse proportion to the number of subtrees uncovered by the current treelet. This has the intuitive appeal of allowing a more thorough exploration of large treelet translation pairs (that are likely to result in better translations) than of smaller, less promising pairs.

#### Pruning treelet translation pairs

Channel model scores and treelet size are powerful predictors of translation quality. Heuristically pruning low scoring treelet translation pairs before the search starts allows the decoder to focus on combinations and orderings of high quality treelet pairs.

- Only keep those treelet translation pairs with an MLE probability above a threshold  $t$ .
- Given a set of treelet translation pairs with identical sources, keep those with an MLE probability within a ratio  $r$  of the best pair.
- At each input node, keep only the top  $k$  treelet translation pairs rooted at that node, as ranked first by size, then by MLE channel model score, then by Model 1 score. The impact of this optimization is explored in Table 5.6.

#### Greedy ordering

The complexity of the ordering step at each node grows with the factorial of the number of children to be ordered. This can be tamed by noting that given a fixed pre- and post-modifier count, our order model is capable of evaluating a single ordering decision independently from other ordering decisions.

One version of the decoder takes advantage of this to severely limit the number of ordering possibilities considered. Instead of considering all interleavings, it considers each potential modifier position in turn, greedily picking the most probable child for that slot, moving on to the next slot, picking the most probable among the remaining children for that slot and so on.

		English	French	English	Japanese
Training	Sentences	500,000		500,000	
	Words	6,598,914	7,234,153	7,909,198	9,379,240
	Vocabulary	72,440	80,758	66,731	68,048
	Singletons	38,037	39,496	50,381	52,911
Test	Sentences	10,000		10,000	
	Words	133,402	153,701	175,655	211,139

**Table 4.1** Data characteristics

The complexity of greedy ordering is linear, but at the cost of a noticeable drop in BLEU score (see Table 5.4). Under default settings our system tries to decode a sentence with exhaustive ordering until a specified timeout, at which point it falls back to greedy ordering.

## 4. Experiments

We evaluated the translation quality of the system using the BLEU metric (Papineni et al., 02) under a variety of configurations. We compared against two radically different types of systems to demonstrate the competitiveness of this approach:

- Pharaoh: A leading phrasal SMT decoder (Koehn et al., 03).
- The MSR-MT system described in Section 1, an EBMT/hybrid MT system.

### 4.1. Language pairs

We ran experiments in English→French and English→Japanese. The latter was chosen deliberately to highlight the challenges facing string-based MT approaches in language pairs with significant word-order differences.

Word order in Japanese is fundamentally very different from English. English is generally SVO (subject first, then verb, then object), where Japanese is SOV with a strong bias for head-final



**Figure 1.** English-Japanese word alignment

structures. Several other differences include:

- Word order is more flexible, since verbal arguments are generally indicated by postpositions, e.g. a direct object is indicated by the postposition を (o), a subject by が (ga).
- Most post-modifying English phrases (such as relative clauses and prepositional phrases) are translated as Japanese pre-modifiers; demonstratives and adjectives remain pre-modifiers.
- Verbal and adjectival morphology in Japanese is relatively complex: information contained in English pre-modifying modals and auxiliaries is often represented as verbal morphology.
- Japanese nouns and noun phrases are not marked for definiteness or number.

The word-aligned sentence pair in Figure 1 demonstrates many of these phenomena.

### 4.2. Data

We used a corpus of Microsoft technical data (e.g., support articles, product documentation) containing over 1 million sentence pairs for each language-pair. We excluded sentences containing XML or HTML tags and for each language pair randomly selected training data sets ranging from 1,000 to 500,000 sentence pairs as well as 10,000 sentences for development testing and parameter tuning, 250 sentences for lambda training and 10,000 sentences for testing. Table 4.1 presents some characteristics of this corpus.

### 4.3. Training

We parsed the source (English) side of the corpus using two different parsers: NLPWIN, a broad-coverage rule-based parser developed at Microsoft Research able to produce syntactic analyses at varying levels of depth (Heidorn, 02)

	English→French, 100K		English→Japanese, 500K	
	BLEU	Sents/min	BLEU	Sents/min
Pharaoh monotone	37.06	4286	25.06	1600
Pharaoh	38.83	162	30.58	82
MSR-MT	35.26	453	-	-
Treelet	40.66	10.1	33.18	21

**Table 5.1** System comparisons

		1k	3k	10k	30k	100k	300k	500K
English → French	Pharaoh	17.20	22.51	27.70	33.73	38.83	42.75	-
	Treelet	18.70	25.39	30.96	35.81	40.66	44.32	-
English → Japanese	Pharaoh	14.85	15.99	18.18	21.89	23.01	26.67	30.58
	Treelet	13.90	15.39	18.94	23.99	25.68	29.97	33.18

**Table 5.2** BLEU scores at different training set sizes, phrase/treelet size 4

and a Treebank parser (Bikel, 04). For the purposes of these experiments we used a dependency tree output with part-of-speech tags and unstemmed, case-normalized surface words.

For word alignment, we used GIZA++, following a standard training regimen of five iterations of Model 1, five iterations of the HMM Model, and five iterations of Model 4, in both directions.

The target language models were trained using only the French and Japanese sides, respectively, of the parallel corpus; additional monolingual data may improve its performance. Finally we trained lambdas via Maximum BLEU (Och, 03) on 250 held-out sentences with a single reference translation, and tuned the decoder optimization parameters ( $n$ -best list size, timeouts etc) on the development test set.

### Pharaoh

The same GIZA++ alignments as above were used in the Pharaoh decoder. We used the heuristic combination described in (Och & Ney, 03) and extracted phrasal translation pairs from this combined alignment as described in (Koehn et al., 03). Except for the order model (Pharaoh uses a penalty on the deviance from monotone), the same models were used: MLE channel model, Model 1 channel model, target language model, phrase count, and word count. Lambdas were trained in the same manner (Och, 03).

### MSR-MT

MSR-MT used its own word alignment approach as described in (Menezes & Richardson, 03) on the same training data. MSR-MT does not use lambdas or a target language model.

## 5. Results

We present BLEU scores on an unseen 10,000 sentence test set using a single reference translation for each sentence. Speed numbers are the end-to-end translation speed in sentences per minute. Unless otherwise specified all results are based on a phrase size of 4 and a training set size of 100,000 sentences for English→French and 500,000 sentences for English→Japanese. Unless otherwise noted all the differences between systems are statistically significant at  $P < 0.01$

Comparative results are presented in Table 5.1. Pharaoh monotone refers to Pharaoh with phrase reordering disabled.

Table 5.2 compares the systems at different training corpus sizes. All the differences are statistically significant at  $P < 0.01$  except for English→Japanese at training set sizes less than 30K. Note that in English→French, where word order differences are mainly local, the gap between the systems narrows slightly with larger corpus sizes, however in English→Japanese, with global ordering differences, the treelet system’s margin over Pharaoh (initially negative) actually increases with increasing corpus size.

Table 5.3 compares Pharaoh and the Treelet system at different phrase sizes. The wide gap at smaller phrase sizes is particularly striking. It appears that while Pharaoh depends heavily on long phrases to encapsulate reordering, our dependency tree-based ordering model enables credible performance even with short phrases/treelets. Our treelet system with two-word treelets outperforms Pharaoh with six-word phrases.

Max size	English→French, 100K		English→Japanese, 100K		English→Japanese, 500K	
	Treelet BLEU	Pharaoh BLEU	Treelet BLEU	Pharaoh BLEU	Treelet BLEU	Pharaoh BLEU
1	37.50	23.18	22.36	12.75	26.95	17.72
2	39.84	32.07	24.53	18.63	31.33	24.30
3	40.36	37.09	25.44	21.37	32.58	28.15
4 (default)	40.66	38.83	25.68	23.01	33.18	30.58
5	40.71	39.41	25.87	23.82	-	-
6	40.74	39.72	25.92	24.43	-	-

**Table 5.3** Effect of maximum treelet/phrase size

		English→French, 100K		English→Japanese, 500K	
		BLEU	Sents/min	BLEU	Sents/min
Monotone	Pharaoh	37.06	4286	25.06	1600
	Treelet with no order model	35.35	39.7	26.43	67
Non-monotone	Pharaoh (default)	38.83	162	30.58	82
	Treelet: greedy ordering	38.85	13.1	31.99	43
	Treelet: exhaustive (default)	40.66	10.1	33.18	21

**Table 5.4** Effect of ordering strategy

Table 5.4 compares different ordering strategies. In contrast to results reported for English-Chinese (Vogel et al., 03), monotone decoding severely degrades the performance of both systems in English→Japanese, presumably due to the large ordering variation between the two languages. In English-French the degradation is less marked.

	BLEU
Pharaoh	23.01
NLPWIN parser: top parse only	25.68
Bikel parser: top parse only	24.15

**Table 5.5** Using different parsers

(English→Japanese, data size 100k, phrase size 4)

Table 5.5 shows the translation results are not dependent on one particular parser, though a a parser trained on a different domain (here, the Treebank) is at a disadvantage.

	BLEU
Pharaoh	30.58
Single NLPWIN parse	33.18
Top 100 NLPWIN parses	34.13
Oracle selection (top 100 NLPWIN parses)	36.91

**Table 5.6** Using multiple parses, parse oracle

(English→Japanese, data size 500k, phrase size 4)

Table 5.6 shows the impact of using the top 100 NLPWIN parses even without any parse scoring. The last line in the table is a parse oracle experiment to explore the potential quality impact of better parse selection – the oracle picks and

translates the one best parse from the top 100 parses.

Table 5.7 is a translation oracle experiment that demonstrates the impact of model error. The oracle picks the translation with the highest BLEU score from among the top N translations produced by the treelet system. Better models may improve performance, though Och et al. (04) suggests achieving this gain this may be difficult.

Number of translations available to oracle	BLEU
1	33.18
4	35.30
16	37.38
64	38.56
256	38.70

**Table 5.7** Translation oracle

(English→Japanese, data size 500k, phrase size 4)

## 5.1. Human Evaluation

Two human raters were presented (in random order) both Pharaoh and Treelet translations of 100 sentences between 10 and 25 words and corresponding source and reference translations. They were asked to pick the more accurate translation. Table 5.8 shows that for most of the sentences, humans prefer the Treelet translations, which is consistent with the BLEU scores above.

		<i>Rater 1</i>			
		<i>Treelet</i>	<i>Neither</i>	<i>Pharaoh</i>	
<i>Rater 2</i>	<i>Treelet</i>	26	21	3	<b>50</b>
	<i>Neither</i>	4	27	3	<b>34</b>
	<i>Pharaoh</i>	0	11	5	<b>16</b>
		<b>30</b>	<b>59</b>	<b>11</b>	

**Table 5. 8** Human evaluation of 100 sentences (English→Japanese, data size 500k, phrase size 4)

## 6. Conclusions and Future Work

We presented a novel approach to syntactically-informed statistical machine translation that leverages a parsed dependency tree representation of the source language via a tree-based ordering model and a syntactically informed decoder. We showed that it outperforms a leading phrasal SMT decoder in BLEU and human quality judgments. We also showed that it outperformed our own logical form-based EBMT/hybrid MT system.

Even in the absence of a parse quality metric, we found that employing multiple parses could improve translation quality. Adding a parse probability may help further the gains from these additional possible analyses.

The syntactic information used in these models is still rather shallow. Order modeling may benefit from additional information such as semantic roles or morphological features. Furthermore, different model structures, machine learning techniques, and target feature representations all have the potential for significant improvements.

## References

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

Aue, Anthony, Arul Menezes, Robert C. Moore, Chris Quirk, and Eric Ringger. Statistical machine translation using labeled semantic dependency graphs. *TMI* 2004.

Charniak, Eugene, Kevin Knight, and Kenji Yamada. Syntax-based language models for statistical machine translation. *MT Summit* 2003.

Cherry, Colin and Dekang Lin. A probability model to improve word alignment. *ACL* 2003.

Chickering, David Maxwell. The WinMine Toolkit. Microsoft Research Technical Report: MSR-TR-2002-103.

Ding, Yuan and Martha Palmer. Automatic learning of parallel dependency treelet pairs. *IJCNLP* 2004.

Heidorn, George. (2000). “Intelligent writing assistance”. In Dale et al. *Handbook of Natural Language Processing*, Marcel Dekker.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. Statistical phrase based translation. *NAACL* 2003.

Lin, Dekang. A path-based transfer model for machine translation. *COLING* 2004.

Menezes, Arul and Stephen D. Richardson. A best-first alignment algorithm for automatic extraction of transfer mappings from bilingual corpora. In *Recent Advances in Example-Based Machine Translation*, M. Carl & A. Way, Eds, Kluwer Academic Publishers, 2003.

Och, Franz Josef and Hermann Ney. A systematic comparison of various statistical alignment models, *Computational Linguistics*, 29(1):19-51, 2003.

Och, Franz Josef. Minimum error rate training in statistical machine translation. *ACL* 2003.

Och, Franz Josef, et al. A smorgasbord of features for statistical machine translation. *HLT/NAACL* 2004.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. *ACL* 2002.

Quirk, Chris, Arul Menezes, and Colin Cherry. Dependency Tree Translation. Microsoft Research Technical Report: MSR-TR-2004-113.

Ringger, Eric, et al. Linguistically informed statistical models of constituent structure for ordering in sentence realization. *COLING* 2004.

Thurmair, Gregor. Comparing rule-based and statistical MT output. *Workshop on the amazing utility of parallel and comparable corpora, LREC*, 2004.

Vogel, Stephan, Ying Zhang, Fei Huang, Alicia Tribble, Ashish Venugopal, Bing Zhao, and Alex Waibel. The CMU statistical machine translation system. *MT Summit* 2003.

Way, A. and N. Gough. Comparing Example-Based and Statistical Machine Translation. *Journal of Natural Language Engineering*, June 2005

Wu, Dekai. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403, 1997.

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