Mach Translat (2006) 20:43–65
DOI 10.1007/s10590-006-9008-4

ORIGINAL PAPER

Dependency treelet translation: the convergence of statistical and example-based machine-translation?

Christopher Quirk · Arul Menezes

Received: 5 January 2006 / Accepted: 25 August 2006 /
Published online: 8 February 2007

Abstract We describe a novel approach to MT 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.

Keywords Example-based machine translation · EBMT · Statistical machine translation · SMT · Syntax · Dependency analysis

1 Introduction

Data-driven Machine Translation approaches, such as Example-based Machine Translation (EBMT) and Statistical Machine Translation (SMT), have revolutionized the field of Machine Translation (MT). Where once the only tractable approach toward MT was writing rule-based transfer systems, we are now seeing the emergence of large-scale translation systems based on data-driven techniques. These approaches are founded on similar datasets but very different principles. However, we believe that that latest generation of data-driven MT systems demonstrate an increasing convergence of EBMT and SMT (cf. Groves and Way 2005). After a brief survey of recent

C. Quirk · A. Menezes
Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
e-mail: chrisq@microsoft.com

A.Menezes (✉)
e-mail: arulm@microsoft.com

developments in EMBT and SMT, we describe a new syntax-based SMT approach that draws successfully from both traditions to produce high-quality translations.

Although it is difficult to pinpoint the exact definition of EBMT (for a more detailed discussion, see Carl 2005; Hutchins 2005; Wu 2005), there are several fundamental principles that are generally accepted by the community as being characteristic of EBMT. First, EBMT is a data-driven approach: translation information is learned primarily from parallel corpora. Second, EBMT relies heavily on the concept of translation by analogy. When presented with a new sentence to translate, an EBMT system attempts to reuse translation information from its parallel corpus, preferably reusing information in segments that are as large as possible. In this sense, EBMT is a generalization of Translation Memory: it goes beyond reuse of translations at the sentence level, attempting more aggressive reuse at the phrase, word, and perhaps morpheme or even character level. Lepage and Denoual (2005) is an example of a particularly pure EBMT system, relying on nothing except a purely analogical assembly mechanism and a parallel corpus. Translation examples are not limited to words or even contiguous phrases; often the most useful units of reuse are discontiguous in nature.

A common way for EBMT systems to exploit complex phenomena is through the use of linguistic analysis. As an example, consider the system of Menezes and Richardson (2003). The first step in training is to parse both source and target language sentences into a deep predicate–argument structure representation, which normalizes away many of the surface differences between languages. Next these deep structures are aligned; from this aligned structure, translation mappings can be automatically extracted. To translate a new sentence, it is first parsed into this deep representation. Then a target-language deep representation is constructed by combining translation mappings learned from the parallel corpus. Finally the target-language sentence is generated from the deep structure using a hand-written or machine-learned generation component. Other systems follow similar lines with some interesting variations. Kurohashi et al. (2005) for instance obviate a target-language generation componnent by employing a shallower analysis of both languages.

At the same time SMT began to develop using the same resources but different ideas. Parallel corpora also formed the foundation of SMT, though initial attempts at statistical translation models were less focused on the idea of reusing examples. Instead, effort was devoted to defining generative models that were sufficiently powerful to accommodate translational divergences while allowing tractable estimation. The touchstone of this early work in SMT is Brown et al. (1993), who define a series of generative models each providing a distribution over the set of foreign-language sentences and word alignments given a source-language sentence. These models were to be used as channel models in a noisy-channel decoder; *n*-gram language models (like those commonly used in speech recognition) could act as the target-language model. However, these initial systems never achieved the speed or quality necessary to break into the mainstream translation market.

The recent activity in SMT has instead been driven by a by-product of the generative models. Along with estimating parameters of channel models, these models could also be used to produce a word alignment. As the EBMT community had recognized for years, identifying word and phrase translations requires a fine-grained correspondence between pieces of a sentence: an accurate word alignment opened many possibilities for SMT. Seeing the potential power of larger translational units, approaches such as alignment templates (Och and Ney 2004) and phrasal SMT (Koehn et al. 2003) took a major step toward EBMT (perhaps an unintentional one) by extracting multiword translations from a word-aligned parallel corpus and stringing them together to form a translation. However these new "phrasal" approaches were still grounded in statistical decision theory. Example phrase pairs with counts were not sufficient to form a full SMT system. The first systems still used something similar to the noisy-channel approach to model translation, and tried to find heuristic search methods that approximated optimal decoding behavior as much as possible. Latter systems generalized the decoding approach to form what are now called hybrid generative–discriminative models, using maximum entropy models (Och and Ney 2002) or direct optimization of error rates (Och 2003) to optimize functions.

With the above developments, one may easily argue that the convergence of EBMT and SMT had already begun. For instance, recent work, including Way and Gough (2005) and Groves and Way (2005) has noted and explored the similarities between phrasal SMT and Marker-based EBMT. Yet these phrasal SMT systems only started to exploit the information that had been used for years in EBMT systems. Even the very first EBMT systems found that effective reuse often requires noncontiguous phrases in either the source language or the target language. Other well-known phenomena (such as boundary friction (Somers 2003)) also pose problems for these systems (though target language models may mitigate this somewhat). Nor does phrasal SMT have an explicit model to account for ordering differences between languages. While arbitrary reordering of words is allowed within memorized phrases, typically only a small amount of phrase reordering is allowed, often modeled in terms of string-level offsets. 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 postmodifying prepositional phrases become Japanese premodifying 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.

A natural solution to many of these problems is the incorporation of syntax. Whether by incorporating linguistic parsers (as in Yamada and Knight 2002; Charniak et al. 2003) or by simply including algorithms motivated by parsing (as in Wu 1997 or Chiang 2005), syntax-based SMT allows the natural incorporation of discontiguous phrases. Furthermore, syntax-based SMT also provides a natural means of modeling constituent reordering.

An early, elegant approach to syntax-based SMT is that of Inversion Transduction Grammars (ITG) (Wu 1997). No overt linguistic information is used; instead, it uses algorithms inspired by parsers. Translation is viewed 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 nonterminal 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. Melamed (2004) generalizes this work, defining parsing algorithms over multitext grammars and demonstrating how these versatile tools can be used in various stages of a syntax-based SMT system.

In a similar vein, the Hiero approach (Chiang 2005) also uses a decoding algorithm motivated by parsing, but incorporates phrasal translations. This is one of the first syntax-based SMT approaches to show significant improvements over a phrasal SMT baseline. There are several important lessons to be drawn from this work. Phrasal translations, effective decoding algorithms, and combinations of statistical models together produce a formidable backbone of an MT system.

Other systems do take advantage of actual linguistic information in translation. Yamada and Knight (2002) employ a parser in the target language to train probabilities on a set of operations that convert an English tree to a Japanese string. These models are then used in a noisy-channel decoder to find the best English translation. Such an approach improves fluency slightly (Charniak et al. 2003), but does not 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. In a similar vein, Graehl and Knight (2004) propose a framework for tree transduction that allows efficient training of transducers using the Expectation-Maximization algorithm (Dempster et al. 1977). This promising approach of training channel models has the potential to improve string-to-tree translation.

Parsers can be also used in the source language. For instance, Lin (2004) 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 (Papineni et al. 2002) are not yet at the level of leading phrasal SMT systems.

Other approaches take advantage of both source and target language parsers. Imamura et al. (2005) use both Japanese and English parsers to help limit the computational complexity and make certain linguistic generalizations more evident. This approach succeeds in outperforming a phrasal SMT baseline in a Japanese-to-English translation task using only a small set of models.

2 Dependency treelet translation

In this paper we propose a novel dependency tree-based approach to phrasal SMT that 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. We believe this approach reinforces the convergence of EBMT and SMT: our translations are firmly grounded in the training data, yet the translation process is guided by a number of probabilistic models in a general log-linear framework.

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 (cf. Langlais and Gotti (2006), whose translation model links source treelets and target strings).

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 and Ney (2003).

This approach offers the following advantages over string-based SMT systems: instead of 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 discontiguous phrases, including combinations such as verb–object, article–noun, adjective–noun, 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 premodifying adjective in English translates into a postmodifier in French.

2.1 Corpus analysis and preparation

The following sections describe the process by which a word-aligned parallel dependency-tree corpus is constructed from sentence-aligned data. These aligned parallel dependency trees are used to train the translation system.

2.1.1 Source analysis

We require only a relatively shallow source-language analysis. We assume that source-language fragments can be part-of-speech tagged, and a simple dependency analysis can be produced. Arc labels are not required. The dependency analysis is viewed in one of two isomorphic ways. First, we can look at this analysis as head annotation: each word in the sentence has a unique parent, except for one word, which is the root of the sentence. This parent function forms a directed acyclic graph. Second, we can view this analysis in a tree-like manner: each word is annotated with a list of its premodifying children and its postmodifying children.

2.1.2 Word alignment

We also require a word alignment of the parallel corpus. A word alignment can be represented as a binary relation "$\sim$" between the source words and the target words in each sentence. The only restriction we place on this relation is that it cannot be many-to-many. That is, if S and T are sets of source and target words such that $s \sim t$ for all $s \in S$ and $t \in T$, then either $|S| = 1$ or $|T| = 1$.

However we currently obtain word alignments with GIZA++ (Och and Ney 2003), which limits the word alignments to involve at most one word in the target side. Therefore, we follow the common practice of deriving many-to-many alignments by running the IBM models in both directions and combining the results heuristically (cf. Groves and Way (2005) for a pictorial representation of this process). Our heuristics differ in that they constrain many-to-one alignments to be contiguous in the source dependency tree. We apply the following rules in order:

1. Accept all alignments from the intersection.
2. Accept all alignments that are unique on both sides (i.e., the only alignment in the union from the given source word is to the given target word, and vice versa).
3. Accept all alignments that are unique on one side (i.e., the only alignment in the union from the given source word is to that target word, but the target word has other non-unique alignments, or vice versa).
4. Accept those many-to-one alignments that are adjacent to existing alignments in the source dependency tree (i.e., accept an alignment (s_i, t_k) if we have already accepted an alignment (s_j, t_k), and either the parent of s_i is s_j, or the parent of s_j is s_i.
5. Accept all one-to-many alignments.

The resulting alignment is a superset of the intersection and a subset of the union.

2.1.3 Dependency projection

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 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[1] of the "many" target words, and make the rest of the target words dependent on it.

Unaligned target words[2] 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 leftmost (or rightmost) word that is aligned. Algorithm 1 provides a pseudocode description of the process.

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 of the phrase pair *startup properties and options* and *propriétés et options de démarrage* (lit. 'properties and options of startup') in Fig. 1 using the word alignment in Fig. 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 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 Fig. 2c, reattaching *démarrage* to *et* suffices to produce the correct order.

[1] If the target language is Japanese, leftmost may be more appropriate.

[2] 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.

Algorithm 1 Tree projection algorithm

```
function ProjectDeptree( $S$ : source nodes, $T$ : target nodes, $A$ : alignment)
  $a \leftarrow$ AlignmentFunction($S$, $T$, $A$)
  $h_t \leftarrow$ ProjectDeptreeBackbone ($\varepsilon$, root($S$), $\varepsilon$, $a$)
  $h_t \leftarrow$ AttachOthers($h_t$, $T$)
  $h_t \leftarrow$ Reattach($t$, $h_t$)
  return $h_t$
function AlignmentFunction($S$, $T$, $A$)
  $a \leftarrow \emptyset$
  for all $s \in S$ do
    $X \leftarrow \{t \in T | (s,t) \in A\}$
    $a(s) \leftarrow$ rightmost word in $X$; $\varepsilon$ if $X = \emptyset$
  return $a$
function ProjectDeptreeBackbone($s_0$, $s_1$, $t_0$, $a$)
  $h \leftarrow \emptyset$
  $t_1 \leftarrow a(s_1)$
  if $t_0 \neq \varepsilon \wedge t_1 \neq \varepsilon$ then
    $h(t_1) \leftarrow t_0$
  for all $s_2 \in$children($s_1$) do
    $h \leftarrow h \cup$ ProjectDeptreeBackbone($s_1$, $s_2$, $t_1$, $a$)
  return $h$
function AttachOthers($T$, $h$, $A$)
  for all $t \in T$ such that $h(t) = \varepsilon$ do
    if $t$ is aligned then
      Find $s \in S$, $t' \in T$ such that $(s,t), (s,t') \in A$ and $h(t') \neq \varepsilon$
      $h(t) = t'$
    else
      Find closest aligned words to the left and right $t_l, t_r$
      $h(t) \leftarrow t_l$ if $t_l$ is further from the root than $t_r$; $h(t) \leftarrow t_r$ otherwise.
  return $h$
function Reattach($t$, $h$)
  $Q \leftarrow \langle t \rangle$; a queue.
  $h' \leftarrow \emptyset$; the new parent relation without crossing dependencies
  while $Q$ is not empty do
    $t_1 \leftarrow$ Pop($Q$)
    $t_0 \leftarrow h(t)$
    while $(t_0, t_1)$ crosses some dependency already in $h'$ do
      $t_0 \leftarrow h'(t_0)$
    $h'(t_1) \leftarrow t_0$
    Enqueue($Q$, children($t$))
  return $h'$
```

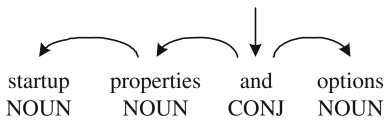
2.1.4 *Extracting treelet translation pairs*

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.

2.2 Models

Generally we view translation as a global search problem: given an input dependency tree, a search component (or *decoder*) produces possible translation candidates, which are scored by a log-linear combination of feature functions. The highest scoring candidate is returned as the final translation.

Fig. 1 An example dependency tree



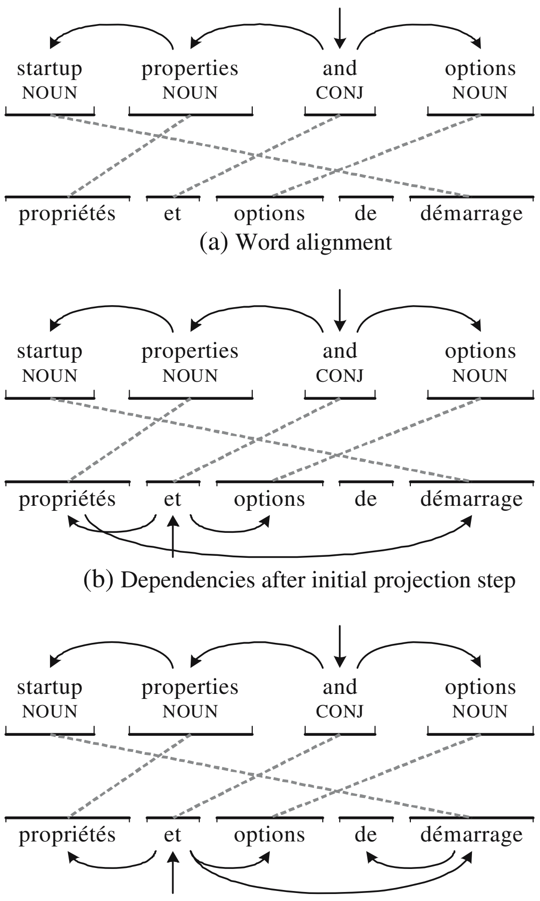
A candidate in this search consists of:

- S : a source dependency tree,
- T : a target dependency tree,
- A : a word alignment between the source and target trees, and
- I : a set of treelet translation pairs that are a partitioning of S and T into treelets.

Put formally, then, we wish to find $\text{argmax}_{T,A,I}\{\textsc{score}(S,T,A,I)\}$. The SCORE function is defined as a log-linear combination of the values of a set of feature functions F (1).

$$\textsc{score}(S,T,A,I) = \sum_{f \in F} w_f \cdot \log f(S,T,A,I) \tag{1}$$

Fig. 2 Dependency trees produced by the algorithm (**a**) Word alignment (**b**) Dependencies after initial projection step (**c**) Dependencies after reattachment



(a) Word alignment

(b) Dependencies after initial projection step

(c) Dependencies after reattachment

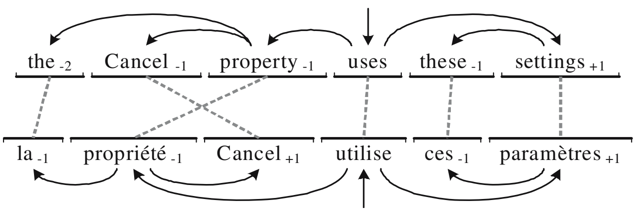


Fig. 3 Aligned dependency tree pair annotated with head-relative positions

Theoretically, these feature functions could be any real-valued function of the candidate. Most, however, are simply the scores from probabilistic models. The following sections explore these models in more detail.

2.2.1 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 as a target phrase in position *j* (Koehn et al. 2003).

We attempt to improve on these approaches by incorporating syntactic information into the ordering process. Our model assigns a probability to the order of a target tree given an unordered target tree, a source tree, and a word alignment between the two. Since we assume that dependencies are projective and phrases cohere during translation (that is, the translation of a source constituent is a target constituent that is a contiguous substring), the location of a constituent within a sentence is determined by the position of its root node relative to its parent and the other modifiers of that parent. One way to indicate this order is with head-relative positions: the closest premodifier of a node has position -1, the next has position -2, and so on; the closest postmodifier of a node has position $+1$, and so on. Figure 3 demonstrates an aligned dependency tree pair annotated with head-relative positions for the sentence pair in (2).

(2) The Cancel property uses these settings.
La propriété Cancel utilise ces paramètres.

We can estimate a conditional probability distribution $\mathbf{Pr}(T|S,Q,A)$ over ordered target trees T given a source tree S, an unordered target tree Q (a tree with a head function, but no relative order between nodes), and the word alignment A between S and T. Then, given a function φ from ordered trees to their unordered counterparts, we can use this probabilistic order model as another feature function in the log-linear combination (3).

$$f_{\text{Order}}(S,T,A,I) = \mathbf{Pr}(T|S,\varphi(T),A) \tag{3}$$

We model this distribution in terms of head-relative orderings, and we make the strong independence assumption that each node is positioned independently (4).

$$\mathbf{Pr}(T|S,Q,A) \approx \prod_{t\in Q} \mathbf{Pr}(\text{pos}(t)|S,Q,A) \tag{4}$$

Furthermore, to make training tractable and to simplify decoding, we use only a very small set of features reflecting local information in the dependency tree to model this probability. In particular, these are the features used:

– the lexical items of the target node and and its parent;
– the lexical items of the source nodes aligned to the target node and its parent;
– the part-of-speech (category, or "cat") of the source nodes aligned to the node and its parent;
– the head-relative position of the source node aligned to the node.

One could also make a Markov assumption and condition on information from the siblings of the target node; we found that this complicated the model and had little impact in practice.

As an example, consider the children of *propriété* in Fig. 3. The head-relative positions of its modifiers *la* and *Cancel* are -1 and $+1$, respectively. Then the model attempts to predict the target positions as described in Fig. 4. Despite all the limitations of this model, it can capture important linguistic generalizations that are not available to purely phrase-based systems. For instance, the generalization that English premodifiers of nouns generally become postmodifiers in French unless they are determiners is straightforward to learn from a decision tree over the given features; similarly, one can see that English prepositions generally become Japanese postpositions.

The training corpus in the form of word-aligned parallel dependency trees acts as a training set for supervised classifiers. From each target-language node, we extract a single data point, where the target feature is the head-relative position in the target tree, and the other features are simply those available in the training set.

Such a training set could be used with many different machine learning methods. We have found decision trees to be surprisingly effective: they are both fast to train, relatively robust, and able to discover feature conjunctions, hence solve problems that are not linearly separable.

For a given test feature vector, we compute a probability distribution from the decision tree by first following the path to the leaf dictated by that feature vector, and then use the MLE (maximum likelihood estimation) over the target feature counts at that leaf (Chickering 2002).

2.2.2 Channel models

We incorporate two distinct channel models, an MLE model and a model computed using Model-1 word-to-word alignment probabilities as in Vogel et al. (2003). The

$$\begin{aligned}
\mathbf{Pr}(\mathrm{pos}(m1) = -1 \mid\ & \mathrm{lex}(m1) = \mathit{la}, \mathrm{lex}(h) = \mathit{propriété},\\
& \mathrm{lex}(\mathrm{src}(m1)) = \mathit{the}, \mathrm{lex}(\mathrm{src}(h) = \mathit{property},\\
& \mathrm{cat}(\mathrm{src}(m1)) = \textsc{det}, \mathrm{cat}(\mathrm{src}(h)) = \textsc{noun},\\
& \mathrm{pos}(\mathrm{src}(m1)) = -2) \bullet\\
\mathbf{Pr}(\mathrm{pos}(m2) = +1 \mid\ & \mathrm{lex}(m2) = \mathit{Cancel}, \mathrm{lex}(h) = \mathit{propriété},\\
& \mathrm{lex}(\mathrm{src}(m2)) = \mathit{Cancel}, \mathrm{lex}(\mathrm{src}(h)) = \mathit{property},\\
& \mathrm{cat}(\mathrm{src}(m2)) = \textsc{noun}, \mathrm{cat}(\mathrm{src}(h)) = \textsc{noun},\\
& \mathrm{pos}(\mathrm{src}(m2)) = -1)
\end{aligned}$$

Fig. 4 Example order model decomposition

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. We have four channel-model feature functions, since a model can either predict target given source (a direct model) (5,7) or source given target (an inverse model) (6,8), and it can be estimated using MLE (5,6) or via lexical translation probabilities (7,8).

$$f_{\text{DirectMLE}}(S,T,A,I) = \prod_{\langle\sigma,\tau\rangle\in I} \frac{c(\sigma,\tau)}{c(\sigma,*)} \tag{5}$$

$$f_{\text{InverseMLE}}(S,T,A,I) = \prod_{\langle\sigma,\tau\rangle\in I} \frac{c(\sigma,\tau)}{c(*,\tau)} \tag{6}$$

$$f_{\text{DirectLex}}(S,T,A,I) = \prod_{\langle\sigma,\tau\rangle\in I} \prod_{t\in\tau} \sum_{s\in\sigma} p(s|t) \tag{7}$$

$$f_{\text{InverseLex}}(S,T,A,I) = \prod_{\langle\sigma,\tau\rangle\in I} \prod_{s\in\sigma} \sum_{t\in\tau} p(t|s) \tag{8}$$

The lexical translation probabilities $p(s|t)$ and $p(t|s)$ used here are from Model 1 of Brown et al. (1993).

2.2.3 *Target-language model*

One crucial component of modern SMT systems is a target-language model. These models appear simple—model the probability of a target string with a Markov assumption, that is, find the probability of each word in the context of the previous n words—but have a major impact on the fluency, grammaticality, and even accuracy of translation. As our dependency trees are ordered, an in-order walk suffices to enumerate the target words in order, so adding an n-gram target-language model as another feature is quite easy (9).

$$f_{\text{Target}}(S,T,A,I) = \prod_{i=1}^{|T|} \mathbf{Pr}\left(t_i|t_{i-n}^{i-1}\right) \tag{9}$$

We estimate the n-gram probabilities using modified Kneser–Ney smoothing (Goodman 2001).

2.2.4 *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 (10). We also add a feature that counts the number of target words; this acts as an insertion/deletion bonus/penalty, and counteracts the preference of the target language model for shorter sentences (11).

$$f_{\text{PhraseCount}}(S,T,A,I) = e^{|I|} \tag{10}$$
$$f_{\text{WordCount}}(S,T,A,I) = e^{|T|} \tag{11}$$

2.3 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 discontiguous or overlapping—and a combinatorially explosive ordering space.

Our decoding approach is influenced by ITG (Wu 1997) 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 dependants of a node to be ordered in any arbitrary manner and use the order model to estimate probabilities. 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'.

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 of each child s', which is computed in the same way on-demand and memoized. 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 Fig. 5 and treelet translation pair in Fig. 6 for the sentence pair in (12).

(12) The software is installed on your computer.
Les logiciels sont installés sur votre ordinateur.

This treelet translation pair covers all the nodes in Fig. 5 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*.

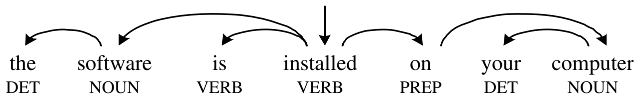
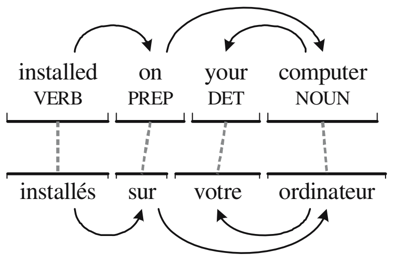


Fig. 5 Example input dependency tree

Fig. 6 Example treelet translation pair



2.3.1 *Dynamic programming*

Converting this exhaustive search to dynamic programming relies on several key observations. First we note that the candidate scores are generally multiplicative. The channel-model probability of a candidate constructed from a treelet and existing candidates for subtrees not covered by that treelet is simply the product of the treelet and candidate channel-model probabilities; a similar situation exists for the word-count and phrase-count features. For the order model, we must also score the head-relative positions of the treelet as well as the root elements of the existing candidates, and for the target-language model, we must multiply in the probabilities of the words at the boundary of each candidate. Second, we note that these additional probabilities depend only on a very small amount of information in the candidate: the order model probability depends only on the lexical item at the root of the tree, and an n-gram target-language model requires only the first and last $(n-1)$ words in each subtree. Therefore, for any given source subtree, we need to keep only the best-scoring translation candidate for each combination of (head, leading $(n-1)$-gram, $(n-1)$-gram bigram). In the worst case, however, this still requires keeping $O(V^5)$ candidates per input subtree, where V is the target language vocabulary size.

Although dynamic programming does limit the search space, it alone is not sufficient to produce a real-time translation system. Therefore, we explore the following optimizations that have proven effective in experimental settings, but do not preserve optimality.

2.3.2 *Beam search*

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 beam size of approximately ten.

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 beam size 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.

2.3.3 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.

– Keep only those treelet translation pairs with an MLE probability above a threshold t. This has the greatest impact on small, common treelets (such as translations of the English word *the*) that have many possible translations. The vast majority of these translations, however, are due to a single sentence pair in which the word alignment was suboptimal.
– 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. This final pruning heuristic has the greatest impact on translation speed.

2.3.4 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 limit severely 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.

The complexity of greedy ordering is linear, but at the cost of a noticeable drop in BLEU score (see results in the next section). 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.

3 Technical data experiments

We evaluated the translation quality of the system using the BLEU metric (Papineni et al. 2002) 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. 2003),
– the MSR-MT system described in Sect. 1, an EBMT/hybrid MT system.

3.1 Language pairs

We ran experiments in translating English to French and English to 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, where Japanese is SOV with a strong bias for head-final structures. Several other differences include:

- Word order is more flexible, since verbal arguments are generally indicated by postpositions, for example, a direct object is indicated by the postposition *o*, a subject by *ga*.
- Many English phrases that are realized as postmodifiers (such as relative clauses and prepositional phrases) are translated as Japanese premodifiers; demonstratives and adjectives remain premodifiers.
- Verbal and adjectival morphology in Japanese is relatively complex: information contained in English premodifying 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 pairs in Figs. 7 and 8 (below) demonstrate many of these phenomena.

3.2 Data

We used a corpus of Microsoft technical data (for example, 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 1 presents basic characteristics of these corpora.

3.3 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 2000), and a Treebank parser (Bikel 2004). 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++ (Och and Ney 2003), 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. Treelets were extracted from this word-aligned parallel corpus; Table2 presents some statistics about those corpora. We note that fewer treelet translation pairs were extracted from the English–Japanese parallel corpus; this is presumably due the difficulty of aligning that language pair.

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 2003) on 250 held-out

Table 1 Data characteristics

		English	French	English	Japanese
Training	Sentences	500,000	500,000	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	10,000	10,000
	Words	133,402	153,701	175,665	211,139

Table 2 Treelet statistics

	English–French, 300k	English–Japanese, 500k
Total treelet translation pairs	42,201,316	35,420,445
Distinct treelet translation pairs	30,188,949	20,600,885
Distinct source treelets	12,659,291	9,554,323

sentences with a single reference translation, and tuned the decoder optimization parameters (beam size, cutoffs, etc.) on the development test set.

3.3.1 Pharaoh

The same GIZA++ alignments as above were used in the Pharaoh decoder. We used the heuristic combination described in Och and Ney (2003) and extracted phrasal translation pairs from this combined alignment as described in Koehn et al. (2003). 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 2003).

3.3.2 MSR-MT

MSR-MT used its own word-alignment approach as described in Menezes and Richardson (2003) on the same training data. MSR-MT does not use lambdas or a target-language model.

3.4 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/treelet size of 4 and a training set size of 100,000 sentences for English to French and 500,000 sentences for English to Japanese. Unless otherwise noted all the differences between systems are statistically significant at $p < 0.01$.

Comparative results are presented in Table 3. "Pharaoh monotone" refers to Pharaoh with phrase reordering disabled. Table 4 compares the systems at different training corpus sizes. All differences are statistically significant at $p < 0.01$ except for English–Japanese at training-set sizes less than 30k. Note that in English–French,

Table 3 System comparisons

	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 4 BLEU scores on training data subsets, phrase/treelet size 4

	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

where word-order differences are mainly local, the gap between the systems narrows slightly with larger corpus sizes, but in English–Japanese, with global ordering differences, the treelet system's margin over Pharaoh (initially negative) actually increases with increasing corpus size.

Table 5 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.

Table 6 presents some statistics on the number of treelets available during decoding time as well as the average number of treelets used in an individual sentence. The number of treelets matched per sentence is slightly less in English–Japanese vs. English–French; this is probably due to the smaller number of treelets extracted overall. However, we note that the number of treelet translation pairs is slightly higher. Although this could be due to a greater amount of ambiguity in English–Japanese translation, we think a more likely explanation is that the lower word-alignment quality leads to less consistent mappings, and therefore more possible translation pairs. Finally, we see that on average more translation pairs are used to translate a single sentence; this may partly account for the lower BLEU scores in English–Japanese.

Table 7 compares different ordering strategies. In contrast to results reported for English–Chinese (Vogel et al. 2003), monotone decoding severely degrades the performance of both systems in English–Japanese. We presume that this is due to the broad differences in word order between the two languages. In English–French the degradation is less marked.

Table 8 shows that the translation results are not dependent on one particular parser, though a parser trained on a different domain (here, the Treebank) is at a disadvantage.

Table 5 Effect of maximum treelet/phrase size measured in BLEU scores

Max. size		English–French		English–Japanese			
		100k		100k		500k	
		Treelet	Pharaoh	Treelet	Pharaoh	Treelet	Pharaoh
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 6 Treelet statistics at decoding time

	English–French	English–Japanese
	300k	500k
Average source treelets matched	61.84	57.24
Average treelet translation pairs available	114.84	130.20
Average treelet translation pairs used	8.44	9.96

Table 7 Effect of ordering strategy, measured by BLEU score

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

Table 8 Using different parsers (English–Japanese, 100k, size 4)

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

Table 9 shows the impact of using parses beyond the 1-best. The first experiment is quite simple: we translate each parse separately, and keep the 1-best translation from each. Finally, we pick the highest-scoring translation from each of these with no additional information about the parse. Even without any features of the parse itself, this approach boosts the BLEU score by a significant amount. This highlights the problems with the pipelined approach of using the single best parse, and suggests that decoding over a packed forest of parses may produce significantly better results.

The last line in Table 9 is an oracle experiment to demonstrate the potential improvements from better parse selection. In this experiment, we translate each parse

Table 9 Using *k*-best parses (English–Japanese, 500k, size 4)

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

separately and again keep only the top-scoring translation for each parse. However, instead of picking among these translations based on score, we consult the reference translation and pick the candidate most like the reference; the improvement in quality is quite impressive. Better parse-ranking mechanisms may, therefore, have a positive impact on translation quality. Unfortunately we do not have source-language sentences with both gold-standard dependency annotations and target-language reference translations, so it is difficult to identify the correlation between monolingual parse accuracy and efficacy in translation.

Returning to the 1-best parse, in Table 10 we see a translation oracle experiment that demonstrates the impact of model error. The oracle picks the translation most like the reference translation from among the top *n* translations produced by the treelet system. Better models have the potential for major quality improvements, though Och et al. (2004) suggest achieving this gain is difficult.

Finally, we expect that the gains due to better parse selection may be somewhat orthogonal to the gains due to better channel-model selection. Often when a parse is incorrect, it precludes selection of the correct translation. In the future we plan to explore the interactions between *k*-best parsing and *n*-best translations in more detail. Discriminative reranking including features from the source parse tree are particularly promising.

Figure 7 shows examples of sentences where the Pharaoh translation was preferred over the Treelet translation, with some analysis, and Fig. 8 shows the opposite case.

Table 10 Translation oracle (English–Japanese, 500k, size 4)

Translations	BLEU
1	33.18
4	35.30
16	37.38
64	38.56
256	38.70

Table 11 Human evaluation of 100 sentences (English–Japanese, 500k, size 4)

	Rater 1	Rater 2
Treelet preferred	30	50
Neither preferred	59	34
Pharaoh preferred	11	16

SOURCE	type a new name for the Riched32.dll file (for example , Riched32.old) , and then press ENTER .
REFERENCE	Riched32.dll の 新しい 名前 (たとえば 、 Riched32.old) を 入力 して 、 Enter キー を 押します 。
PHARAOH	Riched32.dll ファイル (たとえば 、 Riched32.old の 新しい 名前 を 入力 し 、 Enter キー を 押します 。
TREELET	Riched32.dll ファイル (たとえば 、 Riched32.old) 、 の 新しい 名前 を 入力 と し 、 Enter キー を 押します 。
ANALYSIS	*Treelet incorrectly translates "type" as "入力 とし" ("consider as input") rather than "入力し" ("input")*

SOURCE	click to select the Use the Same Proxy Server for All Protocols check box .
REFERENCE	[すべてのプロトコルに同じプロキシサーバーを使用する] チェックボックスをクリックしてオン に します 。
PHARAOH	同じ Proxy Server の すべて の プロトコル を 使用 する] チェック ボックス を オン に します 。
TREELET	クリック 、チェックボックス を すべて の プロトコル に 同じ プロキシ サーバー を 使用 して 選択 します 。
ANALYSIS	*Treelet misparses input sentence: "Use the Same Proxy Server for All Protocols" is not identified as a user interface term modifying "check box". Main verb is therefore incorrect, amongst other errors.*

Fig. 7 Sentences where the Pharaoh translation was preferred over the Treelet translation

3.5 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 11 shows that for most of the sentences, humans prefer the Treelet translations, which is consistent with the BLEU scores above.

4 Conclusions and future work

We have presented a novel approach to syntactically informed SMT by leveraging a parsed dependency-tree representation of the source language via a tree-based ordering model and a syntactically informed decoder. We have shown that it outperforms a leading phrasal SMT decoder in BLEU and human quality judgments. We have also shown 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.

SOURCE	in Visual Studio .NET , create a new Managed C++ application called determine0S .
REFERENCE	Visual Studio .NET で determine0S という 名前 で 新しい Managed C アプリケーション を 作成 します。
PHARAOH	Visual Studio .NET で は 、 新しい Managed C アプリケーション determine0S と 呼ばれます 。
TREELET	Visual Studio .NET で determine0S と 呼ばれる 新しい Managed C アプリケーション を 作成 します。
ANALYSIS	*Pharaoh drops main verb "create"; "call" becomes main verb; "determineOS" incorrectly compounded onto "application"*
SOURCE	in the Named box , type scsi1hlp.vxd , and then click Find Now .
REFERENCE	[ファイルまたはフォルダの名前]ボックスに scsi1hlp.vxd と入力し、[検索 開始] ボタン を クリック します。
PHARAOH	[名前] ボックス に 入力 し 、 [検索 開始] を クリック します。 scsi1hlp.vxd
TREELET	[名前] ボックス で 、 scsi1hlp.vxd 、 入力 し [検索 開始] を クリック します。
ANALYSIS	*Pharaoh incorrectly moves "scsi1hlp.vxd" end of sentence. Treelet translation is not perfect, however: should have placed accusative marker on "scsi1hlp.vxd" and used "に" instead of "で".*
SOURCE	you may be able to recover some disk space by quitting unneeded programs .
REFERENCE	不要 な プログラム を 終了 して 、 ディスク の 空き 領域 を 回復 します。
PHARAOH	一部 の ディスク 領域 を 回復 できる 場合 が あります 不要 な プログラム を 終了 します。
TREELET	不要 な プログラム を 終了 して 一部 の ディスク スペース を 回復 できる 場合 が あります。
ANALYSIS	*Pharaoh directly concatenates clauses with no conjunction.*
SOURCE	when an ALTER TABLE statement is executed , the ROWCOUNT value of the session is taken into account .
REFERENCE	ALTER TABLE ステートメントが実行 されるときに 、その セッションの ROWCOUNT の値が考慮されます 。
PHARAOH	ALTER TABLE ステートメント を 実行 する と 、 セッション が ROWCOUNT の 値 に します 。
TREELET	ALTER TABLE ステートメント が 実行 される と 、 ROWCOUNT の 値 を 、 セッション が 考慮 します 。
ANALYSIS	*Pharaoh loses man verb "taken into account". Again Treelet translation is not perfect: less fluent word order, and the transitive verb has an inanimate subject, which is strongly dispreferred by Japanese speakers.*

Fig. 8 Sentences where the Treelet translation was preferred over the Pharaoh translation

Acknowledgements We would like to thank Colin Cherry for his contributions and insight, Bob Moore for many enlightening technical discussions, and Chris Brockett, Hisami Suzuki, and Takako Aikawa for crucial insights and discussions of Japanese linguistic phenomena.

References

Bikel DM (2004) Intricacies of Collins parsing model. Comput Ling 30:479–511

Brown PF, Della Pietra SA, Della Pietra VJ, Mercer RL (1993) The mathematics of statistical machine translation: parameter estimation. Comput Ling 19:263–311

Carl M (2005) A system-theoretic view of EBMT. Mach Translat 19:229–249

Carl M, Way A (eds) (2003) Recent advances in example-based machine translation, Kluwer Academic Publishers Dordrecht, The Netherlands

Charniak E, Knight K, Yamada K (2003) Syntax-based language models for statistical machine translation. In: MT Summit IX, Proceedings of the ninth machine translation summit, New Orleans, USA, pp 40–46

Chiang D (2005) A hierarchical phrase-based model for statistical machine translation. In: 43rd annual meeting of the Association for Computational Linguistics, Ann Arbor, MI pp 263–270

Chickering DM (2002) The WinMine toolkit. Technical Report MSR-TR-2002-103, Microsoft Research, Seattle, WA

Dempster A, Laird N, Rubin D (1977) Maximum likelihood from incomplete data via the EM algorithm. J Royal Stat Soc 39:1–38

Goodman J (2001) A bit of progress in language modeling. Technical Report MSR-TR-2001-72, Microsoft Research, Seattle, WA

Graehl J, Knight K (2004) Training tree transducers. In: HLT-NAACL 2004: Human language technology conference of North American chapter of the Association for Computational Linguistics, Boston, MA, pp 105–112

Groves D, Way A (2005) Hybrid data-driven models of machine translation. Mach Translat 19:301–323

Heidorn G (2000) Intelligent writing assistance. In: Dale R, Moisl H, Somers H (eds) Handbook of natural language processing. Marcel Dekker New York, NY, pp 181–208

Hutchins J (2005) Example-based machine translation—a review and commentary. Mach Translat 19:197–211

Imamura K, Okuma H, Sumita E (2005) Practical approach to syntax-based statistical machine translation. In: MT Summit X, The tenth machine translation summit, Phuket, Thailand, pp 267–274

Koehn P, Och FJ, Marcu D (2003) Statistical phrase-based translation. In: HLT-NAACL 2003: Human language technology conference of North American chapter of the Association for Computational Linguistics, Edmonton, Alberta, Canada, pp 127–133

Kurohashi S, Nakazawa T, Alexis K, Kawahara D (2005) Example-based machine translation pursuing fully structural NLP. In: Proceedings of the international workshop on spoken language translation, Pittsburgh, PA, pp.207–212

Langlais P, Gotti F (2006) EBMT by tree-phrasing. Mach Translat 20:1–25

Lepage Y, Denoual E (2005) Purest ever example-based machine translation: detailed presentation and assessment. Mach Translat 19:251–282

Lin D (2004) A path-based transfer model for machine translation. In: Coling: 20th international conference on computational linguistics, Geneva, Switzerland, pp 625–630

Melamed ID (2004) Statistical machine translation by parsing. In: 42nd annual meeting of the Association for Computational Linguistics, Barcelona, Spain, pp 653–660

Menezes A, Richardson SD (2003) A best-first alignment algorithm for extraction of transfer mappings. In: Carl and Way (2003), pp 421–442

Och FJ (2003) Minimum error rate training in statistical machine translation. In: 41st annual meeting of the Association for Computational Linguistics, Sapporo, Japan, pp 160–167

Och FJ, Gildea D, Khudanpur S, Sarkar A, Yamada K, Fraser A, Kumar S, Shen L, Smith D, Eng K, Jain V, Jin Z, Radev D (2004) A smorgasbord of features for statistical machine translation. In: HLT-NAACL 2004: Human language technology conference of North American chapter of the Association for Computational Linguistics, Boston, MA, pp 161–168

Och FJ, Ney H (2002) Discriminative training and maximum entropy models for statistical machine translation. In: 40th annual meeting of the Association for Computational Linguistics, Philadelphia, PA, pp 295–302

Och FJ, Ney H (2003) A systematic comparison of various statistical alignment models. Comput Ling 29:19–51

Och FJ, Ney H (2004) The alignment template approach to statistical machine translation. Comput Ling 30:417–449

Papineni K, Roukos S, Ward T, Zhu W-J (2002) Bleu: A method for automatic evaluation of machine translation. In: 40th annual meeting of the Association for Computational Linguistics, Philadelphia, PA, pp 311–318

Somers H (2003) An overview of EBMT. In: Carl and Way (2003), pp 3–58 [Revised version of article in Mach Translat 14 (1999), 113–158]

Vogel S, Zhang Y, Huang F, Tribble A, Venugopal A, Zhao B, Waibel A (2003) The CMU statistical machine translation system. In: MT Summit IX, Proceedings of the ninth machine translation summit, New Orleans, USA, pp 402–409

Way A, Gough N (2005) Comparing example-based and statistical machine translation. Nat Lang Eng 11:295–309

Wu D (1997) Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. Comput Ling 23:377–403

Wu D (2005) MT model space: Statistical vs. compositional vs. example-based machine translation. Mach Translat 19:213–227

Yamada K, Knight K (2002) A decoder for syntax-based statistical MT. In: 40th annual meeting of the Association for Computational Linguistics, Philadelphia, PA, pp 303–310