# Using Dependency Order Templates to Improve Generality in Translation

**Arul Menezes** and **Chris Quirk**
Microsoft Research
One Microsoft Way, Redmond, WA 98052, USA
{arulm, chrisq}@microsoft.com

## Abstract

Today's statistical machine translation systems generalize poorly to new domains. Even small shifts can cause precipitous drops in translation quality. Phrasal systems rely heavily, for both reordering and contextual translation, on long phrases that simply fail to match out-of-domain text. Hierarchical systems attempt to generalize these phrases but their learned rules are subject to severe constraints. Syntactic systems can learn lexicalized and unlexicalized rules, but the joint modeling of lexical choice and reordering can narrow the applicability of learned rules. The treelet approach models reordering separately from lexical choice, using a discriminatively trained order model, which allows treelets to apply broadly, and has shown better generalization to new domains, but suffers a factorially large search space. We introduce a new reordering model based on *dependency order templates*, and show that it outperforms both phrasal and treelet systems on in-domain and out-of-domain text, while limiting the search space.

## 1 Introduction

Modern phrasal SMT systems such as (Koehn et al., 2003) derive much of their power from being able to memorize and use long phrases. Phrases allow for non-compositional translation, local reordering and contextual lexical choice. However the phrases are fully lexicalized, which means they generalize poorly to even slightly out-of-domain text. In an open competition (Koehn & Monz, 2006) systems trained on parliamentary proceedings were tested on text from 'news commentary' web sites, a very slightly different domain. The 9 phrasal systems in the English to Spanish track suffered an absolute drop in BLEU score of between 4.4% and 6.34% (14% to 27% relative). The treelet system of Menezes et al. (2006) fared somewhat better but still suffered an absolute drop of 3.61%.

Clearly there is a need for approaches with greater powers of generalization. There are multiple facets to this issue, including handling of unknown words, new senses of known words etc. In this work, we will focus on the issue of reordering, i.e. can we learn how to transform the sentence structure of one language into the sentence structure of another, in a way that is not tied to a specific domain or sub-domains, or indeed, sequences of individual words.

An early attempt at greater generality in a purely phrasal setting was the alignment template approach (Och & Ney 2004); newer approaches include formally syntactic (Chiang 2005), and linguistically syntactic approaches (Quirk et al. 2005), (Huang et al. 2006). In the next section, we examine these representative approaches to the reordering problem.

## 2 Related Work

Our discussion of related work will be grounded in the following tiny English to Spanish example, where the training set includes:

```
a very old book
un libro más  antiguo
a  book  very old[1]

the old man
el  hombre viejo
the man    old

it is very important
es muy  importante
is very important
```

[1] English gloss of Spanish sentences in italics.



*Proceedings of the Second Workshop on Statistical Machine Translation*, pages 1–8,
Prague, June 2007. 

and the test sentence and reference translation are

```
a very old man
un hombre muy  viejo
a  man    very old
```

Note that while the first training pair has the correct structure for the test sentence, most of the contextually correct lexical choices come from the other two pairs.

### 2.1 Phrasal translation, Alignment templates

The relevant phrases (i.e. those that match the test sentence) extracted from these training pairs are shown in Table 2.1. Only phrases up to size 3 are shown. The ones in italics are 'correct' in that they can lead to the reference translation. Note that none of the multi-word phrases lead to the reference, so the local reordering often captured in the phrasal model is no help at all in ordering this sentence. The system is unable to learn the correct structure from the first sentence because the words are wrong, and from the second sentence even though the phrase *old man* has the right words in the right order, it does not lead to the reference translation because the translation of *very* cannot be inserted in the right place.

| | |
|---|---|
| *a* | *un* |
| very | más |
| old | antiguo |
| very old | más antiguo |
| *old* | *viejo* |
| *man* | *hombre* |
| old man | hombre viejo |
| *very* | *muy* |

**Table 2.1**: Relevant extracted phrases

Looking at this as a sparse data issue we might suspect that generalization could solve the problem. The alignment template approach (Och & Ney, 2004) uses word classes rather than lexical items to model phrase translation. Yet this approach loses the advantage of context-sensitive lexical selection: the word translation model depends only on the word classes to subcategorize for translations, which leads to less accurate lexical choice in practice (Zens & Ney, 2004).

### 2.2 Hierarchical translation

Hierarchical systems (Chiang, 2005) induce a context-free grammar with one non-terminal directly from the parallel corpus, with the advantage of not requiring any additional knowledge source or tools, such as a treebank or a parser. However this can lead to an explosion of rules. In order to make the problem tractable and avoid spurious ambiguity, Chiang restricts the learned rules in several ways. The most problematic of these is that every rule must have at least one pair of aligned words, and that adjacent non-terminals are not permitted on the source side. In Table 2.2 we show the additional hierarchical phrases that would be learned from our training pairs under these restrictions. Again only those applicable to the test sentence are shown and the 'correct' rules, i.e. those that lead to the reference, are italicized.

| | |
|---|---|
| X1 old | X1 antiguo |
| very X1 | más X1 |
| very old X1 | X1 más antiguo |
| X1 old X2 | X2 X1 antiguo |
| very X1 X2 | X2 más X1 |
| *X1 man* | *hombre X1* |
| old X1 | X1 viejo |
| X1 old man | X1 hombre viejo |
| *X1 very* | *X1 muy* |
| *very X2* | *muy X2* |
| X1 very X2 | X1 muy X2 |

**Table 2.2**: Additional hierarchical phrases

Note that even though from the first pair, we learn several rules with the perfect reordering for the test sentence, they do not lead to the reference because they drag along the contextually incorrect lexical choices. From the second pair, we learn a rule (*X1 old man*) that has the right contextual word choice, but does not lead to the reference, because the paucity of the grammar's single non-terminal causes this rule to incorrectly imply that the translation of *very* be placed before *hombre.*

### 2.3 Constituency tree transduction

An alternate approach is to use linguistic information from a parser. Transduction rules between Spanish strings and English trees can be learned from a word-aligned parallel corpus with parse trees on one side (Graehl & Knight, 2004). Such rules can be used to translate from Spanish to English by searching for the best English language tree for a given Spanish language string (Marcu et al., 2006). Alternately English trees produced by a parser can be transduced to

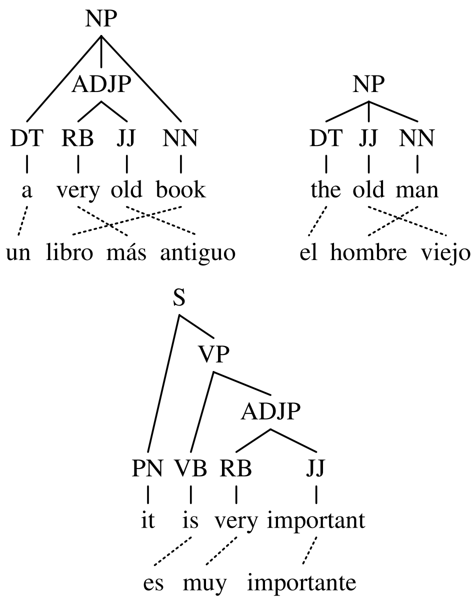


**Figure 2.1**: Constituency parses

Spanish strings using the same rules (Huang et al., 2006). Translation rules may reach beyond one level in the syntax tree; this extended domain of locality allows many phenomena including both lexicalized and unlexicalized rules. However reordering and translation are modeled jointly, which may exacerbate data sparsity. Furthermore it forces the system to pick between unlexicalized rules that capture reordering and lexicalized rules that model context-sensitive translation.

For instance, the following rules can be extracted from the first sentence of the corpus:

$r_1$: **un** x1 x2 → NP(DT(**a**) ADJP:x2 NN:x1)
$r_2$: x1 x2 → ADJP(RB:x1 JJ:x2)

Although together they capture the necessary reordering for our test sentence pair, they do not allow for context sensitive translations of the ambiguous terms *very* and *old*; each must be selected independently. Disappointingly, no single constituency tree transduction rule derived from this corpus translates *old man* as *hombre viejo* in a single step on the test sentence: the syntactic structures are slightly different, but the difference is sufficient to prevent matching. [2] Again we note that phrases provide utility by capturing both reordering and context. While xRS rules provide an elegant and powerful model of reordering, they come with a potential cost in context-sensitive translation.

## 2.4 Dependency treelet translation

We previously described (Quirk et al, 2005) a linguistically syntax-based system that parses the source language, uses word-based alignments to project a target dependency tree, and extracts paired dependency tree fragments (treelets) instead of surface phrases. In contrast to the xRS approach, ordering is very loosely coupled with translation via a separate discriminatively trained dependency tree-based order model. The switch to a dependency parse also changes the conditioning information available for translation: related lexical items are generally adjacent, rather than separated by a path of unlexicalized non-terminals. In effect, by using a looser matching requirement, treelets retain the context-sensitive lexical choice of phrases: treelets must only be a connected subgraph of the input sentence to be applicable; some children may remain uncovered.

Figure 2.2 shows source dependency parses and projected target dependencies for our training data; Figure 2.3 shows the treelet pairs that this system would extract that match the input

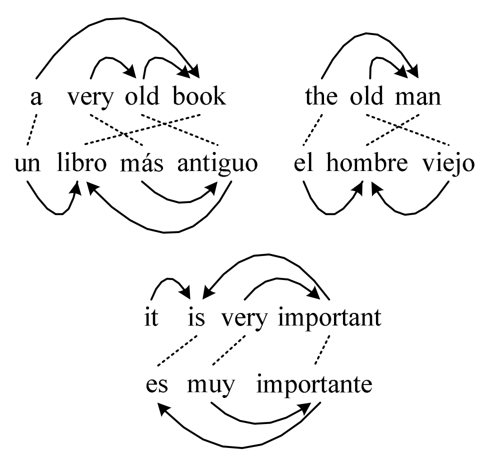


**Figure 2.2**: Dependency trees for training pairs

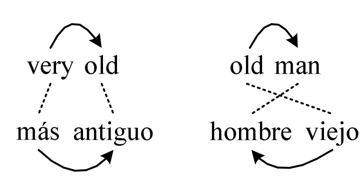


**Figure 2.3**: Relevant extracted treelets

[2] Marcu et al. (2006) and Zollmann et al. (2006) recognize this problem and attempt to alleviate it by grafting surface phrases into constituency trees by various methods.

sentence (treelets of size 1 are not shown). The second treelet supplies the order of *viejo* with respect to its head, and unlike the case with xRS rules, we can use this to make the correct contextual word choice. The difference is that because xRS rules provide *both* reordering and word choice, each rule must match all of the children at any given tree node. On the other hand, treelets are allowed to match more loosely. The translations of the unmatched children (*un* and *muy* in this case) are placed by exploring all possible orderings and scoring them with both order model and language model. Although this effectively decouples lexical selection from ordering, it comes at a huge cost in search space and translation quality may suffer due to search error. However, as mentioned in Section 1, this approach is able to generalize better to out-of-domain data than phrasal approaches. Koehn and Monz (2006) also include a human evaluation, in which this system ranked noticeably higher than one might have predicted from its BLEU score.

## 3 Dependency Order Templates

The Dependency Order Templates approach leverages the power of the xR rule formalism, while avoiding the problems mentioned in Section 2.3, by constructing the rules on the fly from two separately matched components: (a) Dependency treelet translation pairs described in Section 2.4 that capture contextual lexical translations but are underspecified with respect to ordering, and (b) Order templates, which are unlexicalized rules (over dependency, rather than constituency trees) that capture reordering phenomena.

Formally, an order template is an unlexicalized transduction rule mapping dependency trees containing only parts of speech to unlexicalized target language trees (see Figure 4.1b).

Given an input sentence, we combine relevant treelet translation pairs and order templates to construct lexicalized transduction rules for that sentence, and then decode using standard transduction approaches. By keeping lexical and ordering information orthogonal until runtime, we can produce novel transduction rules not seen in the training corpus. This allows greater generalization capabilities than the constituency tree transduction approaches of Section 2.3.

As compared to the treelet approach described in Section 2.4, the generalization capability is somewhat reduced. In the treelet system *all* reorderings are exhaustively evaluated, but the size of the search space necessitates tight pruning, leading to significant search error. By contrast, in the order template approach we consider only reorderings that are captured in some order template. The drastic reduction in search space leads to an overall improvement, not only in decoding speed, but also in translation quality due to reduced search error.

### 3.1 Extracting order templates

For each pair of parallel training sentences, we parse the source sentence, obtain a source dependency tree, and use GIZA++ word alignments to project a target dependency tree as described in Quirk et al. (2005).

Given this pair of aligned source and target dependency trees, we recursively extract one order template for each pair of aligned non-leaf source and target nodes. In the case of multi-word alignments, all contiguous[3] aligned nodes are added to the template. Next we recursively add child nodes as follows: For each node in the template, add all its children. For each such child, if it is aligned, stop recursing, if it is unaligned, recursively add its children.

On each template node we remove the lexical items; we retain the part of speech on the source nodes (we do not use target linguistic features). We also keep node alignment information[4]. The resulting aligned source and target sub-graphs comprise the order template. Figure 4.1b lists the order templates extracted from the training pairs in Figure 2.1 that capture all the patterns necessary to correctly reorder the test sentence.

## 4 Decoding

Decoding is treated as a problem of syntax-directed transduction. Input sentences are segmented into a token stream, annotated with part-of-speech information, and parsed into

[3] If a multi-word alignment is not contiguous in either source or target dependency tree no order template is extracted.
[4] If a source or target node aligns to a tree node outside the template, the template breaks phrasal cohesion and is currently discarded. We intend to address these 'structural divergence' patterns in future work.

unlabeled dependency trees. At each node in the input dependency tree we first find the set of matching treelet pairs: A pair matches if its source side corresponds to a connected subgraph of the input tree. Next we find matching order templates: order templates must also match a connected subgraph of the input tree, but in addition, for each input node, the template must match either all or none of its children [5]. Compatible combinations of treelets and order templates are merged to form xR rules. Finally, we search for the best transduction according to the constructed xR rules as scored by a log-linear combination of models (see Section 5).

### 4.1 Compatibility

A treelet and an order template are considered compatible if the following conditions are met: The treelet and the matching portions of the template must be structurally isomorphic. Every treelet node must match an order template node. Matching nodes must have the same part of speech. Unaligned treelet nodes must match an unaligned template node. Aligned treelet nodes must match aligned template nodes. Nodes that are aligned to each other in the treelet pair must match template nodes that are aligned to each other.

### 4.2 Creating transduction rules

Given a treelet, we can form a set of tree transduction rules as follows. We iterate over each source node *n* in the treelet pair; let *s* be the corresponding node in the input tree (identified during the matching). If, for all children of *s* there is a corresponding child of *n*, then this treelet specifies the placement of all children and no changes are necessary. Otherwise we pick a template that matched at *s* and is compatible with the treelet. The treelet and template are unified to produce an updated rule with variables on the source and target sides for each uncovered child of *s*. When all treelet nodes have been visited, we are left with a transduction rule that specifies the translation of all nodes in the treelet and contains variables that specify the placement of all

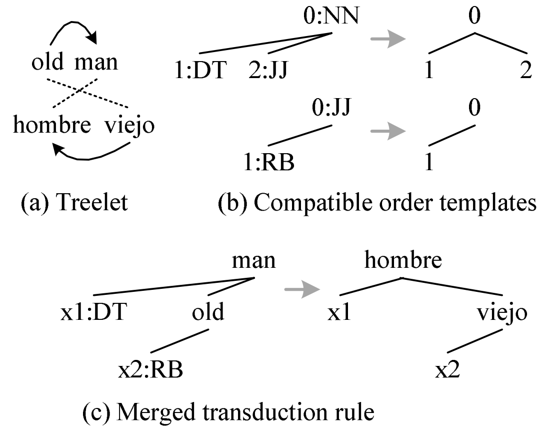


**Figure 4.1**: Merging templates and treelets

uncovered nodes. Due to the independence of ordering and lexical information, we may produce novel transduction rules not seen in the training corpus. Figure 4.1 shows this process as it applies to the test sentence in Section 2.

If, at any node *s,* we cannot find a matching template compatible with the current treelet, we create an artificial *source order template*, which simply preserves the source language order in the target translation. We add a feature function that counts the number of such templates and train its weight during minimum error rate training.

### 4.3 Transduction using xR rules

In the absence of a language model or other contextually dependent features, finding the highest scoring derivation would be a simple dynamic program (Huang et al. 2006) [6].However exact search using an $n$-gram language model leads to split states for each $n$-gram context. Instead we use an approximate beam search moving bottom-up in the tree, much like a CKY parser. Candidates in this search are derivations with respect to the transducer.

Each transduction rule $r$ has a vector of variables $v_{r1}, \dots v_{rk}$. Each variable is associated with an input node $S(v)$. For each input node $s$, we keep a beam of derivations $b[s]$. Derivations are represented as a pair $\langle r, \mathbf{e} \rangle$ where $r$ is a transduction rule and $\mathbf{e} \in \mathbb{N}^k$ is a vector with one integer for each of the $k$ variables in $r$. The interpretation is that the complete candidate can be constructed by recursively substituting for each

[5] This is so the resulting rules fit within the xR formalism. At each node, a rule either fully specifies its ordering, or delegates the translation of the subtree to other rules.

[6] Like Chiang (2005) we only search for the yield of the most likely derivation, rather than the most likely yield.

GetTranslationBeam($s$) // *memoized*
  *prioq* ← ∅
  *beam* ← ∅
  **for** $r \in \mathcal{R}(s)$
    Enqueue(*prioq*, $\langle r, \mathbf{1} \rangle$, EarlyScore($\langle r, \mathbf{1} \rangle$))
  **while** Size(*prioq*) $> 0$
    $\langle r, \mathbf{e} \rangle$ ← PopBest(*prioq*)
    AddToBeam(*beam*, $\langle r, \mathbf{e} \rangle$, TrueScore($\langle r, \mathbf{e} \rangle$))
    **for** $i$ in $1..|\mathbf{e}|$
      Enqueue(*prioq*, $\langle r, \mathbf{e} + \mathbf{1}_i \rangle$,
        EarlyScore($\langle r, \mathbf{e} + \mathbf{1}_i \rangle$))
  **return** *beam*

EarlyScore($\langle r, \mathbf{e} \rangle$)
  $c$ ← RuleScore($r$)
  **for** $i$ in $1..|\mathbf{e}|$
    $s$ ← InputNode(GetVariable ($r$, $i$))
    beam ← GetTranslationBeam($s$)
    $c$ ← $c$ +TrueScore(GetNthEntry(*beam*, $e_i$))
  **return** $c$

**Figure 4.2**: Beam tree transduction

$v_{ri} \in v_{r1} \dots v_{rk}$ the candidate constructed from the $e_i^{\text{th}}$ entry in the beam $b[S(v_{ri})]$.

Figure 4.2 describes the transduction process. Since we approach decoding as xR transduction, the process is identical to that of constituency-based algorithms (e.g. Huang and Chiang, 2007). There are several free parameters to tune:

- Beam size – Maximum number of candidates per input node (in this paper we use 100)
- Beam threshold – maximum range of scores between top and bottom scoring candidate (we use a logprob difference of 30)
- Maximum combinations considered – To bound search time, we can stop after a specified number of elements are popped off the priority queue (we use 5000)

# 5 Models

We use all of the Treelet models we described in Quirk et al. (2005) namely:

- Treelet table with translation probabilities estimated using maximum likelihood, with absolute discounting.
- Discriminative tree-based order model.
- Forward and backward lexical weighting, using Model-1 translation probabilities.
- Trigram language model using modified Kneser-Ney smoothing.
- Word and phrase count feature functions.

In addition, we introduce the following:

- Order template table, with template probabilities estimated using maximum likelihood, with absolute discounting.
- A feature function that counts the number of artificial *source order templates* (see below) used in a candidate.

The models are combined in a log-linear framework, with weights trained using minimum error rate training to optimize the BLEU score.

# 6 Experiments

We evaluated the translation quality of the system using the BLEU metric (Papineni et al., 2002). We compared our system to Pharaoh, a leading phrasal SMT decoder (Koehn et al., 2003), and our treelet system. We report numbers for English to Spanish.

## 6.1 Data

We used the Europarl corpus provided by the NAACL 2006 Statistical Machine Translation workshop. The target language model was trained using only the target side of the parallel corpus. The larger monolingual corpus was not utilized. The corpus consists of European Parliament proceedings, 730,740 parallel sentence pairs of English-Spanish, amounting to about 15M words in each language. The test data consists of 2000 sentences each of development (*dev*), development-test (*devtest)* and test data (*test)* from the same domain. There is also a separate set of 1064 test sentences (*NC-test)* gathered from "news commentary" web sites.

## 6.2 Training

We parsed the source (English) side of the corpus using NLPWIN, a broad-coverage rule-based parser able to produce syntactic analyses at varying levels of depth (Heidorn, 2002). 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++, under a training regimen of five iterations of Model 1, five iterations of HMM, and five iterations of Model 4, in both directions. The forward and backward alignments were symmetrized using a tree-based heuristic combination. The word alignments and English dependency tree were used to project a target tree. From the aligned tree pairs we extracted a treelet table and an order template table.

The comparison treelet system was identical except that no order template model was used.

The comparison phrasal system was constructed using the same GIZA++ alignments and the heuristic combination described in (Och & Ney, 2003). Except for the order models (Pharaoh uses a penalty on the deviance from monotone), the same models were used.

All systems used a treelet or phrase size of 7 and a trigram language model. Model weights were trained separately for all 3 systems using minimum error rate training to maximize BLEU (Och, 2003) on the development set (*dev*). Some decoder pruning parameters were tuned on the development test (*devtest*). The *test* and *NC-test* data sets were not used until final tests.

## 7 Results

We present the results of our system comparisons in Table 7.1 and Figure 7.1 using three different test sets: The in-domain development test data (*devtest*), the in-domain blind test data (*test*) and the out-of-domain news commentary test data (*NC-test*). All differences (except phrasal vs. template on *devtest*), are statistically significant at the p>=0.99 level under the bootstrap resampling test. Note that while the systems are quite comparable on the in-domain data, on the out-of-domain data the phrasal system's performance drops precipitously, whereas the performance of the treelet and order template systems drops much less, outperforming the phrasal system by 2.7% and 3.46% absolute BLEU.

| | *devtest* | *test* | *NC-test* |
|---|---|---|---|
| Phrasal | 0.2910 | 0.2935 | 0.2354 |
| Treelet | 0.2819 | 0.2981 | 0.2624 |
| Template | 0.2896 | 0.3045 | 0.2700 |

**Table 7.1**: System Comparisons across domains

Further insight may be had by comparing the recall[7] for different n-gram orders (Table 7.2). The phrasal system suffers a greater decline in the higher order n-grams than the treelet and template

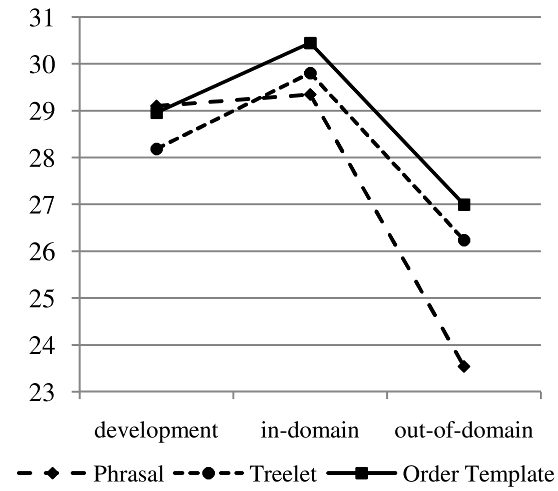


**Figure 7.1**: In-domain vs. Out-of-domain BLEU

systems, indicating that latter show improved generality in reordering.

| | | 1gm | 2gm | 3gm | 4gm |
|---|---|---|---|---|---|
| *Test* | Phrasal | 0.61 | 0.35 | 0.23 | 0.15 |
| | treelet | 0.62 | 0.36 | 0.23 | 0.15 |
| | template | 0.62 | 0.36 | 0.24 | 0.16 |
| *NC-test* | phrasal | 0.58 | 0.30 | 0.17 | 0.10 |
| | treelet | 0.60 | 0.33 | 0.20 | 0.12 |
| | template | 0.61 | 0.34 | 0.20 | 0.13 |

**Table 7.2**: n-gram recall across domains

### 7.1 Treelet vs. Template systems

As described in Section 3.1, the order templates restrict the broad reordering space of the treelet system. Although in theory this might exclude reorderings necessary for some translations, Table 7.3 shows that in practice, the drastic search space reduction allows the decoder to explore a wider beam and more rules, leading to reduced search error *and* increased translation speed. (The *topK* parameter is the number of phrases explored for each span, or rules/treelets for each input node.)

| | *Devtest* BLEU | Sents. per sec |
|---|---|---|
| Pharaoh, beam=100, topK=20 | 0.2910 | 0.94 |
| Treelet, beam=12, topK=5 | 0.2819 | 0.21 |
| Template, beam=100, topK=20 | 0.2896 | 0.56 |

**Table 7.3**: Performance comparisons

Besides the search space restriction, the other significant change in the template system is to include MLE template probabilities as an

[7] n-gram precision cannot be directly compared across output from different systems due to different levels of 'brevity'

additional feature function. Given that the template system operates over rules where the ordering is fully specified, and that most tree transduction systems use MLE rule probabilities to model both lexical selection and reordering, one might ask if the treelet system's discriminatively trained order model is now redundant. In Table 7.4 we see that this is not the case.[8] (Differences are significant at p>=0.99.)

| | *devtest* | *test* | *NC-test* |
|---|---|---|---|
| MLE model only | 0.2769 | 0.2922 | 0.2512 |
| Discriminative and MLE models | 0.2896 | 0.3045 | 0.2700 |

**Table 7.4**: Templates and discriminative order model

Finally we examine the role of frequency thresholds in gathering templates. In Table 7.5 it may be seen that discarding singletons reduces the table size by a factor of 5 and improves translation speed with negligible degradation in quality.

| | *devtest* BLEU | Number of templates | Sentences per sec. |
|---|---|---|---|
| No threshold | 0.2898 | 752,165 | 0.40 |
| Threshold=1 | 0.2896 | 137,584 | 0.56 |

**Table 7.5**: Effect of template count cutoffs

## 8 Conclusions and Future Work

We introduced a new model of Dependency Order Templates that provides for separation of lexical choice and reordering knowledge, thus allowing for greater generality than the phrasal and xRS approaches, while drastically limiting the search space as compared to the treelet approach. We showed BLEU improvements over phrasal of over 1% in-domain and nearly 3.5% out-of-domain. As compared to the treelet approach we showed an improvement of about 0.5%, but a speedup of nearly 3x, despite loosening pruning parameters.

Extraposition and long distance movement still pose a serious challenge to syntax-based machine translation systems. Most of the today's search algorithms assume phrasal cohesion. Even if our search algorithms could accommodate such movement, we don't have appropriate models to account for such phenomena. Our system already extracts extraposition templates, which are a step in the right direction, but may prove too sparse and brittle to account for the range of phenomena.

[8] We speculate that other systems using transducers with MLE probabilities may also benefit from additional reordering models.

## References


Chiang, David. A hierarchical phrase-based model for statistical machine translation. ACL 2005.

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? HLT-NAACL 2004.

Graehl, Jonathan and Kevin Knight. Training Tree Transducers. NAACL 2004.

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

Huang, Liang, Kevin Knight, and Aravind Joshi. Statistical Syntax-Directed Translation with Extended Domain of Locality. AMTA 2006

Huang, Liang and David Chiang. Faster Algorithms for Decoding with Integrated Language Models. ACL 2007 (to appear)

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

Koehn, Philipp and Christof Monz. Manual and automatic evaluation of machine translation between european languages. Workshop on Machine Translation, NAACL 2006.

Marcu, Daniel, Wei Wang, Abdessamad Echihabi, and Kevin Knight. SPMT: Statistical Machine Translation with Syntactified Target Language Phrases. EMNLP-2006.

Menezes, Arul, Kristina Toutanova and Chris Quirk. Microsoft Research Treelet translation system: NAACL 2006 Europarl evaluation. Workshop on Machine Translation, NAACL 2006

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 and Hermann Ney: The Alignment Template Approach to Statistical Machine Translation. Computational Linguistics 30 (4): 417-449 (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: Syntactically informed phrasal SMT. ACL 2005

Zens, Richard and Hermann Ney. Improvements in phrase-based statistical machine translation. HLT-NAACL 2004