Phrase-Based SMT with Shallow Tree-Phrases

Philippe Langlais and Fabrizio Gotti

RALI – DIRO

Université de Montréal,
C.P. 6128 Succ. Centre-Ville
H3C 3J7, Montréal, Canada

{felipe,gottif}@iro.umontreal.ca

Abstract

In this article, we present a translation system which builds translations by gluing together Tree-Phrases, i.e. associations between simple syntactic dependency treelets in a source language and their corresponding phrases in a target language. The Tree-Phrases we use in this study are syntactically informed and present the advantage of gathering source and target material whose words do not have to be adjacent. We show that the phrase-based translation engine we implemented benefits from Tree-Phrases.

1 Introduction

Phrase-based machine translation is now a popular paradigm. It has the advantage of naturally capturing local reorderings and is shown to outperform word-based machine translation (Koehn et al., 2003). The underlying unit (a pair of phrases), however, does not handle well languages with very different word orders and fails to derive generalizations from the training corpus.

Several alternatives have been recently proposed to tackle some of these weaknesses. (Matusov et al., 2005) propose to reorder the source text in order to mimic the target word order, and then let a phrase-based model do what it is good at. (Simard et al., 2005) detail an approach where the standard phrases are extended to account for "gaps" either on the target or source side. They show that this representation has the potential to better exploit the training corpus and to nicely handle differences such as negations in French and English that are poorly handled by standard phrase-based models.

Others are considering translation as a synchronous parsing process e.g. (Melamed, 2004; Ding and Palmer, 2005)) and several algorithms have been proposed to learn the underlying production rule probabilities (Graehl and Knight, 2004; Ding and Palmer, 2004). (Chiang, 2005) proposes an heuristic way of acquiring context free transfer rules that significantly improves upon a standard phrase-based model.

As mentioned in (Ding and Palmer, 2005), most of these approaches require some assumptions on the level of isomorphism (lexical and/or structural) between two languages. In this work, we consider a simple kind of unit: a Tree-Phrase (TP), a combination of a fully lexicalized treelet (TL) and an elastic phrase (EP), the tokens of which may be in non-contiguous positions. TPs capture some syntactic information between two languages and can easily be merged with standard phrase-based engines.

A TP can be seen as a simplification of the treelet pairs manipulated in (Quirk et al., 2005). In particular, we do not address the issue of projecting a source treelet into a target one, but take the bet that collecting (without structure) the target words associated with the words encoded in the nodes of a treelet will suffice to allow translation. This set of target words is what we call an elastic phrase.

We show that these units lead to (modest) improvements in translation quality as measured by automatic metrics. We conducted all our experiments

on an in-house version of the French-English Canadian Hansards.

This paper is organized as follows. We first define a Tree-Phrase in Section 2, the unit with which we built our system. Then, we describe in Section 3 the phrase-based MT decoder that we designed to handle TPs. We report in Section 4 the experiments we conducted combining standard phrase pairs and TPs. We discuss this work in Section 5 and then conclude in Section 6.

2 Tree-Phrases

We call *tree-phrase* (TP) a bilingual unit consisting of a source, fully-lexicalized *treelet* (TL) and a target *phrase* (EP), that is, the target words associated with the nodes of the treelet, in order. A treelet can be an arbitrary, fully-lexicalized subtree of the parse tree associated with a source sentence. A phrase can be an arbitrary sequence of words. This includes the standard notion of phrase, popular with phrased-based SMT (Koehn et al., 2003; Vogel et al., 2003) as well as sequences of words that contain gaps (possibly of arbitrary size).

In this study, we collected a repository of tree-phrases using a robust syntactic parser called SYNTEX (Bourigault and Fabre, 2000). SYNTEX identifies syntactic dependency relations between words. It takes as input a text processed by the TREETAGGER part-of-speech tagger.[1] An example of the output SYNTEX produces for the source (French) sentence "on a demandé des crédits fédéraux" (request for federal funding) is presented in Figure 1.

We parsed with SYNTEX the source (French) part of our training bitext (see Section 4.1). From this material, we extracted all dependency subtrees of depth 1 from the complete dependency trees found by SYNTEX. An elastic phrase is simply the list of tokens aligned with the words of the corresponding treelet as well as the respective offsets at which they were found in the target sentence (the first token of an elastic phrase always has an offset of 0).

For instance, the two treelets in Figure 2 will be collected out of the parse tree in Figure 1, yielding 2 tree-phrases. Note that the TLs as well as the EPs might not be contiguous as is for instance the case with the first pair of structures listed in the example.

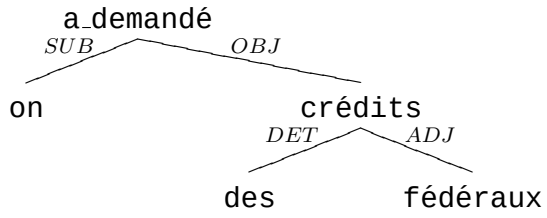


Figure 1: Parse of the sentence "on a demandé des crédits fédéraux" (request for federal funding). Note that the 2 words "a" and "demandé" (literally "have" and "asked") from the original sentence have been merged together by SYNTEX to form a single token. These tokens are the ones we use in this study.

3 The Translation Engine

We built a translation engine very similar to the statistical phrase-based engine PHARAOH described in (Koehn, 2004) that we extended to use tree-phrases. Not only does our decoder differ from PHARAOH by using TPs, it also uses direct translation models. We know from (Och and Ney, 2002) that not using the noisy-channel approach does not impact the quality of the translation produced.

3.1 The maximization setting

For a source sentence f, our engine incrementally generates a set of translation hypotheses $\mathcal{H}$ by combining tree-phrase (TP) units and phrase-phrase (PP) units.[2] We define a hypothesis in this set as $h = \{U_i \equiv (F_i, E_i)\}_{i\in[1,u]}$, a set of u pairs of source (F_i) and target sequences (E_i) of n_i and m_i words respectively:

$$\begin{aligned} F_i &\equiv \{f_{j_n^i} : j_n^i \in [1, |f|]\}_{n\in[1,n_i]} \\ E_i &\equiv \{e_{l_m^i} : l_m^i \in [1, |e|]\}_{m\in[1,m_i]} \end{aligned}$$

under the constraints that for all $i \in [1, u]$, $j_n^i < j_{n+1}^i \,, \forall n \in [1, n_i[$ for a source *treelet* (similar constraints apply on the target side), and $j_{n+1}^i = j_n^i + 1 \,, \forall n \in [1, n_i[$ for a source *phrase*. The way the hypotheses are built imposes additional constraints between units that will be described in Section 3.3. Note that, at decoding time, $|e|$, the number of words

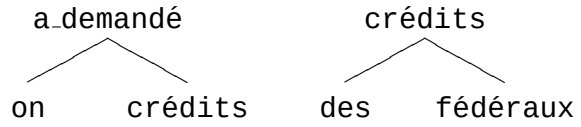
[1] www.ims.uni-stuttgart.de/projekte/corplex/.

[2] What we call here a phrase-phrase unit is simply a pair of source/target sequences of words.

alignment:

a_demandé ≡ request for, fédéraux ≡ federal,
crédits ≡ funding

treelets:



tree-phrases:

```
TL⋆  {{on@-1} a_demandé {crédits@2}}
EP⋆  |request@0||for@1||funding@3|

TL   {{des@-1} crédits {fédéraux@1}}
EP   |federal@0||funding@1|
```

Figure 2: The Tree-Phrases collected out of the SYNTEX parse for the sentence pair of Figure 1. Non-contiguous structures are marked with a star. Each dependent node of a given governor token is displayed as a list surrounding the governor node, e.g. {governor {right-dependent}}. Along with the tokens of each node, we present their respective offset (the governor/root node has the offset 0 by definition). The format we use to represent the treelets is similar to the one proposed in (Quirk et al., 2005).

of the translation is unknown, but is bounded according to $|f|$ (in our case, $|e|_{max} = 2 \times |f| + 5$).

We define the source and target projection of a hypothesis h by the $proj$ operator which collects *in order* the words of a hypothesis along one language:

$$proj_F(h) = \left\{ f_p : p \in \bigcup_{i=1}^{u} \{j_n^i\}_{n \in [1,n_i]} \right\}$$
$$proj_E(h) = \left\{ e_p : p \in \bigcup_{i=1}^{u} \{l_m^i\}_{m \in [1,m_i]} \right\}$$

If we denote by $\mathcal{H}_f$ the set of hypotheses that have f as a source projection (that is, $\mathcal{H}_f = \{h : proj_F(h) \equiv f\}$), then our translation engine seeks $\hat{e} = proj_E(\hat{h})$ where:

$$\hat{h} = \underset{h \in \mathcal{H}_f}{\operatorname{argmax}}\, s(h)$$

The function we seek to maximize $s(h)$ is a log-linear combination of 9 components, and might be better understood as the numerator of a maximum entropy model popular in several statistical MT systems (Och and Ney, 2002; Bertoldi et al., 2004; Zens and Ney, 2004; Simard et al., 2005; Quirk et al., 2005). The components are the so-called feature functions (described below) and the weighting coefficients (λ) are the parameters of the model:

$$\begin{aligned} s(h) = \; & \lambda_{pp_{rf}} \log p_{pp_{rf}}(h) + \lambda_p |h| + \\ & \lambda_{tp_{rf}} \log p_{tp_{rf}}(h) + \lambda_t |h| + \\ & \lambda_{pp_{ibm}} \log p_{pp_{ibm}}(h) + \\ & \lambda_{tp_{ibm}} \log p_{tp_{ibm}}(h) + \\ & \lambda_{lm} \log p_{lm}(proj_E(h)) + \\ & \lambda_d \, d(h) + \lambda_w |proj_E(h)| \end{aligned}$$

3.2 The components of the scoring function

We briefly enumerate the features used in this study.

Translation models Even if a tree-phrase is a generalization of a standard phrase-phrase unit, for investigation purposes, we differentiate in our MT system between two kinds of models: a TP-based model p_{tp} and a phrase-phrase model p_{pp}. Both rely on conditional distributions whose parameters are learned over a corpus. Thus, each model is assigned its own weighting coefficient, allowing the tuning process to bias the engine toward a special kind of unit (TP or PP).

We have, for $k \in \{\text{rf}, \text{ibm}\}$:

$$\begin{aligned} p_{pp_k}(h) &= \textstyle\prod_{i=1}^{u} p_{pp}(E_i|F_i) \\ p_{tp_k}(h) &= \textstyle\prod_{i=1}^{u} p_{tp}(E_i|F_i) \end{aligned}$$

with $p_{\bullet_{rf}}$ standing for a model trained by relative frequency, whereas $p_{\bullet_{ibm}}$ designates a non-normalized score computed by an IBM model-1 translation model p, where f_0 designates the so-called NULL word:

$$p_{\bullet_{ibm}}(E_i|F_i) = \prod_{m=1}^{m_i} \sum_{n=1}^{n_i} p(e_{l_m^i}|f_{j_n^i}) + p(e_{k_m^i}|f_0)$$

Note that by setting $\lambda_{tp_{rf}}$ and $\lambda_{tp_{ibm}}$ to zero, we revert back to a standard phrase-based translation engine. This will serve as a reference system in the experiments reported (see Section 4).

The language model Following a standard practice, we use a trigram target language model $p_{lm}(proj_E(h))$ to control the fluency of the translation produced. See Section 3.3 for technical subtleties related to their use in our engine.

Distortion model d This feature is very similar to the one described in (Koehn, 2004) and only depends on the offsets of the source units. The only difference here arises when TPs are used to build a translation hypothesis:

$$d(h) = -\sum_{i=1}^{n} abs(1 + \overline{F}_{i-1} - \underline{F}_i)$$

where:

$$\begin{aligned} \overline{F}_i &= \begin{cases} \sum_{n \in [1,n_i]} j_n^i / n_i & \text{if } F_i \text{ is a treelet} \\ j_{n_i}^i & \text{otherwise} \end{cases} \\ \underline{F}_i &= j_1^i \end{aligned}$$

This score encourages the decoder to produce a monotonous translation, unless the language model strongly privileges the opposite.

Global bias features Finally, three simple features help control the translation produced. Each TP (resp. PP) unit used to produce a hypothesis receives a fixed weight λ_t (resp. λ_p). This allows the introduction of an artificial bias favoring either PPs or TPs during decoding. Each target word produced is furthermore given a so-called word penalty λ_w which provides a weak way of controlling the preference of the decoder for long or short translations.

3.3 The search procedure

The search procedure is described by the algorithm in Figure 3. The first stage of the search consists in collecting all the units (TPs or PPs) whose source part matches the source sentence f. We call U the set of those matching units.

In this study, we apply a simple match policy that we call *exact match* policy. A TL t matches a source sentence f if its root matches f at a source position denoted r and if all the other words w of t satisfy:

$$f_{o_w + r} = w$$

where o_w designates the offset of w in t.

Hypotheses are built synchronously along with the target side (by appending the target material to the right of the translation being produced) by progressively covering the positions of the source sentence f being translated.

Require: a source sentence f
  $U \leftarrow \{u : \text{s-match}(u, f)\}$
  FUTURECOST(U)
  for $s \leftarrow 1$ to $|f|$ **do**
    $S[s] \leftarrow \emptyset$

  $S[0] \leftarrow \{(\emptyset, \epsilon, 0)\}$
  for $s \leftarrow 0$ to $|f| - 1$ **do**
    PRUNE($S[s], \beta$)
    for all hypotheses alive $h \in S[s]$ **do**
      for all $u \in U$ **do**
        if EXTENDS(u, h) **then**
          $h' \leftarrow$ UPDATE(u, h)
          $k \leftarrow |proj_F(h')|$
          $S[k] \leftarrow S[k] \cup \{h'\}$

  return $\text{argmax}_{h \in S[|f|]} \rho : h \rightarrow (p_s, t, \rho)$

Figure 3: The search algorithm. The symbol $\leftarrow$ is used in place of assignments, while $\rightarrow$ denotes unification (as in languages such as Prolog).

The search space is organized into a set S of $|f|$ stacks, where a stack $S[s]$ ($s \in [1, |f|]$) contains all the hypotheses covering exactly s source words. A hypothesis $h = (p_s, t, \rho)$ is composed of its target material t, the source positions covered p_s as well as its score ρ. The search space is initialized with an empty hypothesis: $S[0] = \{(\emptyset, \epsilon, 0)\}$.

The search procedure consists in extending each partial hypothesis h with every unit that can continue it. This process ends when all partial hypotheses have been expanded. The translation returned is the best one contained in $S[|f|]$:

$$\hat{e} = proj_E(\underset{h \in S[|f|]}{\text{argmax}}\, \rho : h \rightarrow (p_s, t, \rho))$$

PRUNE — In order to make the search tractable, each stack $S[s]$ is pruned before being expanded. Only the hypotheses whose scores are within a fraction (controlled by a meta-parameter β which typically is 0.0001 in our experiments) of the score of the best hypothesis in that stack are considered for expansion. We also limit the number of hypotheses maintained in a given stack to the top `maxStack` ones (`maxStack` is typically set to 500).

Because beam-pruning tends to promote in a stack partial hypotheses that translate easy parts (i.e. parts

that are highly scored by the translation and language models), the score considered while pruning not only involves the cost of a partial hypothesis so far, but also an estimation of the future cost that will be incurred by fully expanding it.

FUTURECOST — We followed the heuristic described in (Koehn, 2004), which consists in computing for each source range $[i, j]$ the minimum cost $c(i, j)$ with which we can translate the source sequence f_i^j. This is pre-computed efficiently at an early stage of the decoding (second line of the algorithm in Figure 3) by a bottom-up dynamic programming scheme relying on the following recursion:

$$c(i,j) = min \begin{cases} min_{k \in [i,j[} c(i,k) + c(k,j) \\ min_{u \in U/u_s \cap f_i^j = u_s} score(u_s) \end{cases}$$

where u_s stands for the projection of u on the target side ($u_s \equiv proj_E(u)$), and $score(u)$ is computed by considering the language model and the translation components p_{pp} of the $s(h)$ score. The future cost of h is then computed by summing the cost $c(i, j)$ of all its empty source ranges $[i, j]$.

EXTENDS — When we simply deal with standard (contiguous) phrases, extending a hypothesis h by a unit u basically requires that the source positions of u be empty in h. Then, the target material of u is appended to the current hypothesis h.

Because we work with treelets here, things are a little more intricate. Conceptually, we are confronted with the construction of a (partial) source dependency tree while collecting the target material in order. Therefore, the decoder needs to check whether a given TL (the source part of u) is compatible with the TLs belonging to h. Since we decided in this study to use depth-one treelets, we consider that two TLs are *compatible* if either they do not share any source word, or, if they do, this shared word must be the governor of one TL and a dependent in the other TL.

So, for instance, in the case of Figure 2, the two treelets are deemed compatible (they obviously should be since they both belong to the same original parse tree) because `crédit` is the governor in the right-hand treelet while being the dependent in the left-hand one. On the other hand, the two treelets in Figure 4 are not, since `président` is the governor of both treelets, even though `mr. le président suppléant` would be a valid source phrase. Note that it might be the case that the treelet `{{mr.@-2} {le@-1} président {suppléant@1}}` has been observed during training, in which case it will compete with the treelets in Figure 2.

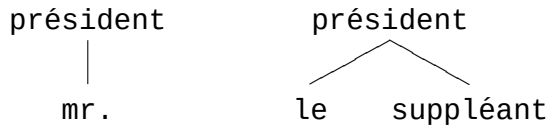


Figure 4: Example of two incompatible treelets. `mr. speaker` and `the acting speaker` are their respective English translations.

Therefore, extending a hypothesis containing a treelet with a new treelet consists in merging the two treelets (if they are compatible) and combining the target material accordingly. This operation is more complicated than in a standard phrase-based decoder since we allow gaps on the target side as well. Moreover, the target material of two compatible treelets may intersect. This is for instance the case for the two TPs in Figure 2 where the word `funding` is common to both phrases.

UPDATE — Whenever u extends h, we add a new hypothesis h' in the corresponding stack $S[|proj_F(h')|]$. Its score is computed by adding to that of h the score of each component involved in $s(h)$. For all but the one language model component, this is straightforward. However, care must be taken to update the language model score since the target material of u does not come necessarily right after that of h as would be the case if we only manipulated PP units.

Figure 5 illustrates the kind of bookkeeping required. In practice, the target material of a hypothesis is encoded as a vector of triplets $\{\langle w_i, \log p_{lm}(w_i|c_i), l_i\rangle\}_{i \in [1,|e|_{max}]}$ where w_i is the word at position i in the translation, $\log p_{lm}(w_i|c_i)$ is its score as given by the language model, c_i denotes the largest conditioning context possible, and l_i indicates the length (in words) of c_i (0 means a unigram probability, 1 a bigram probability and 2 a trigram probability). This vector is updated at each

extension.

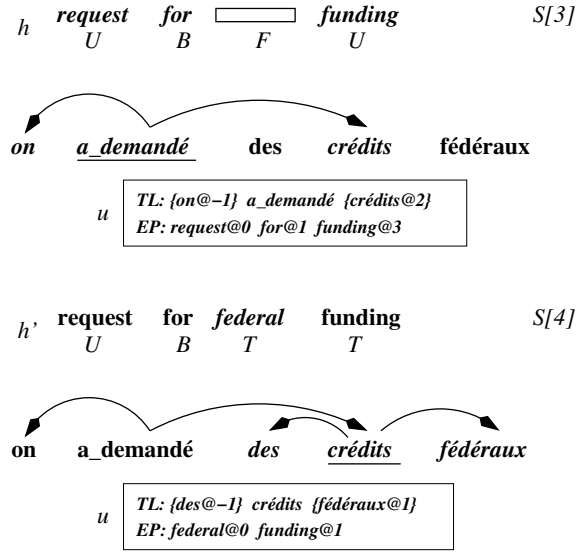


Figure 5: Illustration of the language model updates that must be made when a new target unit (circles with arrows represent dependency links) extends an existing hypothesis (rectangles). The tag inside each occupied target position shows whether this word has been scored by a **U**nigram, a **B**igram or a **T**rigram probability.

4 Experimental Setting

4.1 Corpora

We conducted our experiments on an in-house version of the Canadian Hansards focussing on the translation of French into English. The split of this material into train, development and test corpora is detailed in Table 1. The TEST corpus is subdivided in 16 (disjoints) slices of 500 sentences each that we translated separately. The vocabulary is atypically large since some tokens are being merged by SYNTEX, such as `étaient#financées` (`were financed` in English).

The training corpus has been aligned at the word level by two Viterbi word-alignments (*French2English* and *English2French*) that we combined in a heuristic way similar to the *refined* method described in (Och and Ney, 2003). The parameters of the word models (IBM model 2) were trained with the GIZA++ package (Och and Ney, 2000).

	TRAIN	DEV	TEST
sentences	1 699 592	500	8000
e-toks	27 717 389	8 160	129 601
f-toks	30 425 066	8 946	143 237
e-toks/sent	16.3 (± 9.0)	16.3 (± 9.1)	16.2 (± 9.1)
f-toks/sent	17.9 (± 9.5)	17.9 (± 9.5)	17.9 (± 9.4)
e-types	164 255	2 224	12 143
f-types	210 085	2 481	14 484
e-hapax	68 506	1 469	6 673
f-hapax	90 747	1 704	8 381

Table 1: Main characteristics of the corpora used in this study. For each language *l*, *l*-toks is the number of tokens, *l*-toks/sent is the average number of tokens per sentence (± the standard deviation), *l*-types is the number of different token forms and *l*-hapax is the number of tokens that appear only once in the corpus.

4.2 Models

Tree-phrases Out of 1.7 million pairs of sentences, we collected more than 3 million different kinds of TLs from which we projected 6.5 million different kinds of EPs. Slightly less than half of the treelets are contiguous ones (i.e. involving a sequence of adjacent words); 40% of the EPs are contiguous. When the respective frequency of each TL or EP is factored in, we have approximately 11 million TLs and 10 million EPs. Roughly half of the treelets collected have exactly two dependents (three word long treelets).

Since the word alignment of non-contiguous phrases is likely to be less accurate than the alignment of adjacent word sequences, we further filter the repository of TPs by keeping the most likely EPs for each TL according to an estimate of $p(EP|TL)$ that do not take into account the offsets of the EP or the TL.

PP-model We collected the PP parameters by simply reading the alignment matrices resulting from the word alignment, in a way similar to the one described in (Koehn et al., 2003). We use an in-house tool to collect pairs of phrases of up to 8 words. Freely available packages such as THOT (Ortiz-Martínez et al., 2005) could be used as well for that purpose.

Language model We trained a Kneser-Ney trigram language model using the SRILM toolkit (Stolcke, 2002).

4.3 Protocol

We compared the performances of two versions of our engine: one which employs TPs ans PPs (TP-ENGINE hereafter), and one which only uses PPs (PP-ENGINE). We translated the 16 disjoint subcorpora of the TEST corpus with and without TPs.

We measure the quality of the translation produced with three automatic metrics. Two error rates: the sentence error rate (SER) and the word error rate (WER) that we seek to minimize, and BLEU (Papineni et al., 2002), that we seek to maximize. This last metric was computed with the `multi-bleu.perl` script available at `www.statmt.org/wmt06/shared-task/`.

We separately tuned both systems on the DEV corpus by applying a brute force strategy, i.e. by sampling uniformly the range of each parameter (λ) and picking the configuration which led to the best BLEU score. This strategy is inelegant, but in early experiments we conducted, we found better configurations this way than by applying the Simplex method with multiple starting points. The tuning roughly takes 24 hours of computation on a cluster of 16 computers clocked at 3 GHz, but, in practice, we found that one hour of computation is sufficient to get a configuration whose performances, while subobptimal, are close enough to the best one reachable by an exhaustive search.

Both configurations were set up to avoid distortions exceeding 3 (`maxDist = 3`). Stacks were allowed to contain no more than 500 hypotheses (`maxStack = 500`) and we further restrained the number of hypotheses considered by keeping for each matching unit (treelet or phrase) the 5 best ranked target associations. This setting has been fixed experimentally on the DEV corpus.

4.4 Results

The scores for the 16 slices of the test corpus are reported in Table 2. TP-ENGINE shows slightly better figures for all metrics.

For each system and for each metric, we had 16 scores (from each of the 16 slices of the test corpus) and were therefore able to test the statistical significance of the difference between the TP-ENGINE and PP-ENGINE using a Wilcoxon signed-rank test for paired samples. This test showed that the difference observed between the two systems is significant at the 95% probability level for BLEU and significant at the 99% level for WER and SER.

Engine	WER%	SER%	BLEU%
PP	52.80 ± 1.2	94.32 ± 0.9	29.95 ± 1.2
TP	51.98 ± 1.2	92.83 ± 1.3	30.47 ± 1.4

Table 2: Median WER, SER and BLEU scores ($\pm$ value range) of the translations produced by the two engines on a test set of 16 disjoint corpora of 500 sentences each. The figures reported are percentages.

On the DEV corpus, we measured that, on average, each source sentence is covered by 39 TPs (their source part, naturally), yielding a source coverage of approximately 70%. In contrast, the average number of covering PPs per sentence is 233.

5 Discussion

On a comparable test set (Canadian Hansard texts), (Simard et al., 2005) report improvements by adding non-contiguous bi-phrases to their engine without requiring a parser at all. At the same time, they also report negative results when adding non-contiguous phrases computed from the refined alignment technique that we used here.

Although the results are not directly comparable, (Quirk et al., 2005) report much larger improvements over a phrase-based statistical engine with their translation engine that employs a source parser. The fact that we consider only depth-one treelets in this work, coupled with the absence of any particular treelet projection algorithm (which prevents us from training a syntactically motivated reordering model as they do) are other possible explanations for the modest yet significant improvements we observe in this study.

6 Conclusion

We presented a pilot study aimed at appreciating the potential of Tree-Phrases as base units for example-based machine translation.

We developed a translation engine which makes use of tree-phrases on top of pairs of source/target sequences of words. The experiments we conducted suggest that TPs have the potential to improve translation quality, although the improvements we measured are modest, yet statistically significant.

We considered only one simple form of tree in this study: depth-one subtrees. We plan to test our engine on a repository of treelets of arbitrary depth. In theory, there is not much to change in our engine to account for such units and it would offer an alternative to the system proposed recently by (Liu et al., 2005), which performs translations by recycling a collection of tree-string-correspondence (TSC) examples.

References

Nicola Bertoldi, Roldano Cattoni, Mauro Cettolo, and Marcello Federico. 2004. The ITC-irst statistical machine translation system for IWSLT-2004. In *IWSLT*, pages 51–58, Kyoto, Japan.

Didier Bourigault and Cécile Fabre. 2000. Approche linguistique pour l'analyse syntaxique de corpus. *Cahiers de Grammaire*, (25):131–151. Toulouse le Mirail.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *43rd ACL*, pages 263–270, Ann Arbor, Michigan, USA.

Yuang Ding and Martha Palmer. 2004. Automatic learning of parallel dependency treelet pairs. In *Proceedings of the first International Joint Conference on NLP*.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *43rd ACL*, pages 541–548, Ann Arbor, Michigan, June.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *HLT-NAACL 2004*, pages 105–112, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT*, pages 127–133.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based SMT. In *Proceedings of AMTA*, pages 115–124.

Zhanyi Liu, Haifeng Wang, and Hua Wu. 2005. Example-based machine translation based on tsc and statistical generation. In *Proceedings of MT Summit X*, pages 25–32, Phuket, Thailand.

Evgeny Matusov, Stephan Kanthak, and Hermann Ney. 2005. Efficient statistical machine translation with constraint reordering. In *10th EAMT*, pages 181–188, Budapest, Hongary, May 30-31.

I. Dan Melamed. 2004. Statistical machine translation by parsing. In *42nd ACL*, pages 653–660, Barcelona, Spain.

Franz Joseph Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of ACL*, pages 440–447, Hongkong, China.

Franz Joseph Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the ACL*, pages 295–302.

Franz Joseph Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29:19–51.

Daniel Ortiz-Martínez, Ismael Garciá-Varea, and Francisco Casacuberta. 2005. Thot: a toolkit to train phrase-based statistical translation models. In *Proceedings of MT Summit X*, pages 141–148, Phuket, Thailand, Sep.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *40th ACL*, pages 311–318, Philadelphia, Pennsylvania.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *43rd ACL*, pages 271–279, Ann Arbor, Michigan, June.

Michel Simard, Nicola Cancedda, Bruno Cavestro, Marc Dymetman, Eric Gaussier, Cyril Goutte, Kenji Yamada, Philippe Langlais, and Arne Mauser. 2005. Translating with non-contiguous phrases. In *HLT/EMNLP*, pages 755–762, Vancouver, British Columbia, Canada, Oct.

Andreas Stolcke. 2002. Srilm - an Extensible Language Modeling Toolkit. In *Proceedings of ICSLP*, Denver, Colorado, Sept.

Stephan Vogel, Ying Zhang, Fei Huang, Alicai Tribble, Ashish Venugopal, Bing Zao, and Alex Waibel. 2003. The CMU Statistical Machine Translation System. In *Machine Translation Summit IX*, New Orleans, Louisina, USA, Sep.

Richard Zens and Hermann Ney. 2004. Improvements in phrase-based statistical machine translation. In *Proceedings of the HLT/NAACL*, pages 257–264, Boston, MA, May.