

EBMT by Tree-Phrasing: a Pilot Study

Philippe Langlais[†], Fabrizio Gotti[†], Didier Bourigault[‡] and Claude Coulombe

[†]Univ. de Montréal
Succursale Centre-Ville
H3C 3J7 Montréal
Canada
rali.iro.umontreal.ca

[‡]Univ. de Toulouse-Le Mirail
5, allées Antonio Machado
F-31058 Toulouse Cedex 9
France
bourigault@univ-tlse2.fr

Lingua Technologies Inc.
555, Côte-des-Neiges
H3T 2A9 Montréal
Canada
www.linguatechnologies.com

Abstract

We present a study we conducted to build a repository storing associations between simple dependency treelets in a source language and their corresponding phrases in a target language. To assess the impact of this resource in EBMT, we used the repository to compute coverage statistics on a test bitext and on a n-best list of translation candidates produced by a standard phrase-based decoder.

1 Introduction

Phrase-based machine translation is nowadays a popular paradigm. It has the advantage of naturally capturing local reordering 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 generalise well upon the training corpus.

Several alternatives have been 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. Hildebrand et al. (2005) show that it is possible to adapt the transfer table of a phrase-based model to the specificity of the text being translated. 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 generalise the training corpus and to nicely handle differences such as negations in French and English that are poorly handled by standard phrase-based models.

In this work, we consider a new kind of unit: a Tree-Phrase (TP), a combination of a treelet (TL) and a elastic phrase (EP), the tokens of which may be in non-contiguous positions. Several authors have used treelets as a prime unit to do translation (Gildea, 2003; Ding and Palmer, 2004; Quirk et al., 2005), but mostly with the

idea of projecting a source treelet into its target counterpart.

In this study, we do not address the issue of projecting a 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 handle translation. This set of target words is what we call an elastic phrase (EP). An elastic phrase is not only possibly a non-contiguous sequence of words, but also has the characteristic of having “gaps” of arbitrary size, which is not the case for the phrases considered by Simard et al. (2005).

The objective of this study is to show whether a memory populated with TPs can be of help in a translation task. We are in the early stages of this study and, at this time, do not have a full-fledged decoder using these units. For this reason, in this pilot study, we resorted first to compute coverage statistics of a Tree-Phrase memory on a test bitext and then made post-processing experiments on a n-best list produced by a classic phrase-based decoder. Arguably, if we can show (1) that our Tree-Phrases can cover much of the material to be translated as well as a reference translation, and (2) that these coverage statistics can be correlated with indicators of translation quality, then a memory populated with these units may have some interesting potential.

In order to answer these questions, we conducted the following experiment on the French-English Canadian Hansards. We first parsed the French material with a dependency parser called SYNTAX (Bourigault and Fabre, 2000) which will be briefly presented in Section 2. We collected from this parsed material a set of depth-one treelets that we associated with their target EPs, using a word alignment we computed offline. The main characteristics of this memory are reported in Section 3. Then, we computed several coverage statistics of this memory on a test bitext, employing different

pattern-matching methods. This is reported in Section 4. Finally, we use these coverage statistics in a translation context in Section 5.

2 Syntax

SYNTEX (Bourigault and Fabre, 2000) is a robust and efficient syntactic parser allowing the identification of syntactic dependency relations between words, as well as the extraction of nominal, adjectival and verbal phrases from a corpus. SYNTEX further builds a directed acyclic graph from these phrases, linked to each other by head or expansion relations. Two versions of this software have been created: one for English and one for French.

SYNTEX takes as input a text processed by TREETAGGER¹, a part-of-speech tagger developed at the University of Stuttgart. Some pre- and post-processing of the results from TREETAGGER are made, and through a pipeline of modules of syntactic relation recognition, SYNTEX outputs a number of dependency relations for each sentence.

Currently, the main relation types identified by this tool are subject, direct object, prepositional complement, adjectival modifier, and subordination. Each dependency relation identifies two words: one that acts as a governor, and another one that is its dependent. Each recognition module is “handcrafted” by linguists using the Perl language, and relies on grammatical knowledge and many heuristics to scan a sentence from a candidate governor to find its dependent (or vice-versa), using information from the previous modules.

For example, given the French source sentence “on a demandé des crédits fédéraux” (request for federal funding), SYNTEX outputs several dependency links that we can represent by the structure in Figure 1, where a root node contains the word governing the words of all its child nodes, which are called its dependents. The syntactic dependency relation is presented to the right of the dependent word. Note, however, that we do not consider this information in this work. In this study, SYNTEX was also used to segment sentences into individual tokens, as can be seen in the example in Figure 1.

An example of the output of SYNTEX for the English counterpart of our running example (“request for federal funding”) is shown in Figure 2.

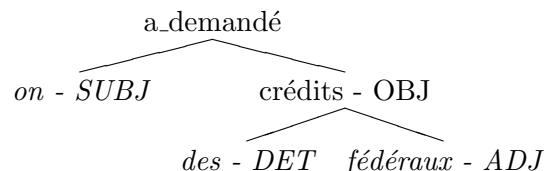


Figure 1: Parse of the sentence “on a demandé des crédits fédéraux” (request for federal funding). Note that the 2 words “a” and “demandé” (literally “have” and “asked”) from the original sentence have been merged together by SYNTEX to form a single token. These tokens are the ones we use in this study.

SRC	Request for federal funding
SYNTEX	NOUN?s request Request 1 0 PREP;2 PREP for for 2 PREP;1 NOUNPREP;4 ADJ federal federal 3 ADJ;4 0 NOUN?s funding funding 4 NOUNPREP;2 ADJ;3

Figure 2: An example of output from SYNTEX. Each line corresponds to a single SYNTEX token. Some tags have been translated in English to facilitate reading.

3 The Memory

We parsed with SYNTEX the source (French) part of our training bitext, that is, about 1.7 million sentences. From this material, we extracted all dependency subtrees of depth 1 from the complete dependency trees found by SYNTEX. For instance, the two treelets in Figure 3 will be collected out of the parse tree in Figure 1.

Prior to that, the full training corpus was aligned at the word level by the method described in (Simard and Langlais, 2003) which recursively splits in two parts both the source and target sentences and allows either a left-to-right alignment (the first part of the source sentence is aligned to the first part of the target sentence, the second parts are aligned together), or an inverted one (the first source part is aligned to the second target one and vice-versa). The best split found at each step is kept and we further split the two parts until we cannot split anymore (that is, when there is at most one token in one side). The computation of the quality of a split is done using a linear combination of two word models (one for each direction) that have been trained on the same training material. We used an IBM model 2 (Brown et al., 1993) for that purpose, whose parameters were

¹<http://www.ims.uni-stuttgart.de/projekte/complex/>.

trained with the GIZA package (Och and Ney, 2000).

An illustration of the output of this alignment procedure is provided for the running example in Figure 3. Once both the word alignment and the treelets are computed, populating the memory with tree-phrases is just a matter of collecting them, and keeping their count over the total training corpus. The format we use to represent the treelets (see Figure 3) is similar to the one proposed in (Quirk et al., 2005): the left and right dependents of a given governor word are listed in order in two separate lists along with their respective offset (the governor/root token always has the offset 0). An elastic phrase is simply the list of tokens aligned to the words of the corresponding treelet as well as the respective offsets at which they were found in the target sentence, relative to the first token position. Note that TLs as well as the EPs might not be contiguous as is for instance the case with the first pair of structures listed in Figure 3.

alignment: a.demandé \equiv request for, fédéraux \equiv federal, crédits \equiv 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 3: The Tree-Phrases collected out of the SYNTAX parse for the sentence pair of Figure 1. Non-contiguous structures are marked by a star.

The tree-phrases (TPs) are stored in a database, whose main characteristics are reported in Table 1. 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 (that is 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 roughly 11 million TLs and 10 million EPs.

We also observe that, as the treelet and the

s	$ treelet $	%-c	$ EP $	%-c
2	639 922	56.8	1 993 896	46.0
3	1 534 468	42.2	3 140 364	38.5
4	737 637	50.5	1 278 254	34.6
5	127 410	53.1	166 465	30.4
6	9 396	36.2	10 108	22.1
7	394	25.9	403	15.4
8	13	0.00	13	7.7
all	3 049 240	47.7	6 589 503	39.8

Table 1: Main statistics measured on the memory as a function of the structure size s . %-c stands for the percentage of structures (TL or EP) that are contiguous. The size of a structure corresponds to the number of tokens it contains. The figures presented here correspond to the number of the different kinds of structures populating the memory and does not account for their respective frequency.

phrase sizes increase, the number of those that are contiguous drops, something that is to be expected.

The 5 most frequent tree-phrases as well as examples of very large ones are reported in Table 2. We note that the most frequent tree-phrases are contiguous ones that would have been captured as well by a “standard” phrase-based model.

4 Coverage Analysis

Rationale One way to get an idea of the exhaustiveness of the memory is to compute coverage statistics on a parallel test corpus disjoint from the training one. This will at least give us an idea of how many translation units a hypothetical TP-based decoder would be able to find for a sentence to be translated. A weak source coverage would be disappointing in our case. Moreover, by computing the coverage of the target (reference) sentence with the target material associated with the source treelets found in the previous step, we get a sense of how meaningful the associations stored in the memory are.

We can also evaluate the respective contributions of contiguous and non-contiguous units to that coverage. To do so, we randomly selected 1 000 pairs of parallel sentences from a subset of the Canadian Hansards not included in our training corpus and tried to match them against the units in our database using various matching methods.

Frequent Tree-phrases

freq	treelet	corresponding EP
75 051	{{{monsieur@-2} {Le@-1} président}}	Mr@0 . @1 Speaker@2
32 601	{{{Le@-1} gouvernement}}	the@0 Government@1
26 347	{{{de@-2} {les@-1} voix}}	Some@0 Honourable@1 Members@2
14 515	{{{Le@-1} ministre}}	the@0 Minister@1
13 043	{{{Madame@-2}{la@-1}Présidente}}	Madam@0 Speaker@1

Long Tree-phrases

TL	treelet	corresponding EP
8	{{{par@-3} {un@-2} {excellent@-1} Chili {con@1} {carne@2} {servi@3} {Léger@6}}}	culmination@0 a@2 Chili@3 con@4 carne@5 feast@6 provided@7 Leger@11
8	{{{sur@-2} {la@-1} question {fondamentale@1} {de@2} {à@8} {nationale@14} {de@15}}}	and@0 on@2 fundamental@4 point@5 to@11 national @ 15

Table 2: The 5 most frequent tree-phrases acquired and 2 examples of especially long ones.

Notation We describe here the notation we will use for the coverage analysis. Let S be a source (French) sentence, with n tokens $s_1 \dots s_n$. Let E be a target (English) sentence, with m tokens $e_1 \dots e_m$.

We also define $t_1 \dots t_k$ to be the tokens of the treelet T . o_1, \dots, o_k are their associated offsets (recall that the root of the treelet has an offset of 0). We call r the token index in S at which T is rooted. It follows that $s_r = \text{root token of } T$.

4.1 Match Policies

We experimented with various matching methods between treelets and source sentences and between elastic phrases and target sentences. All of these methods share a criterion: to have a match, the words in the treelet or elastic phrase must be in the same order as those found in the source/target sentence. No token reordering is allowed.

Source match policies (s-match) For source treelets, we devised an *exact* (E) and a *relaxed* (R) match policy. We say that the treelet T *exactly* matches S if:

$$\forall i \in [1, k], e_i \equiv s_{r+o_i}$$

For the relaxed policy, all the tokens of T must be found in S , but the offsets constraint is relaxed. That is, to match a treelet T to a sentence S , we must find a strictly monotonous function $f : [1, k] \rightarrow [1, n]$, such that:

$$h \rightarrow r \quad \text{where } o_h \equiv r \\ \forall i \in [1, k], e_i \equiv s_{f(i)}$$

Target match policies (t-match) When a treelet matches, its corresponding phrases are retrieved from the memory and matched against the target sentence E . We experimented with three different match policies for phrases. For all these match methods, the search starts at the beginning of E , i.e. at e_1 .

With the *exact* (E) match method, we consider that we have a match when we find the phrase verbatim in the target sentence, with the same gaps between each token.

With the *relaxed* (R) method, we have a match if the tokens of the phrase are encountered in the same order in the target sentence, regardless of their offsets. This latter method allows the tokens of a phrase that are only separated by, say, 2 tokens to match a sentence where they become separated by 18 tokens. This goes against our intuition that the word gaps in non-contiguous phrases must not be stretched beyond a certain limit.

We therefore added a third method *relaxed with stretch limit* (R+S), similar to the second one, where we limit the “elasticity” of those gaps to a maximum of 3 times their original size.

4.2 Upper-Bound Coverage

We used the algorithm shown in Figure 4 to compute various source and target coverage fig-

ures. The idea is simple. We proceed in two steps. First, we find the set tl of all treelets **s-matching** the source sentence S . Then, for each treelet T in tl , we find all corresponding elastic phrases which **t-match** the target sentence E . The positions in S and E at which these pairs of corresponding treelet-phrases match are finally marked as “covered” by the algorithm. Any position s_i or e_j may therefore be covered by many units. This is why this algorithm gives us an upper-bound coverage. We will refine the idea of coverage in Subsection 4.3.

```

for all source tokens  $s_i$  of the sentence  $S$  do
  let  $tl$  be the set of TLs with root token  $s_i$ 
  for all  $T \in tl$  do
    if  $T$  s-matches  $S$  then
      let  $ep$  be the set of EPs associated with
       $T$  in the repository
      for all  $p \in ep$  do
        if  $p$  t-matches the target sentence
         $E$  then
          mark the match positions of  $T$ 
          and  $p$  in  $S$  and  $E$  as covered

```

Figure 4: Algorithm to compute the source and target coverage. The two matching functions **s-match** and **t-match** are discussed in the text.

Results Table 3 shows the results we gathered using the six possible combinations of these policies on 1000 parallel pairs of sentences, corresponding to 17798 source tokens and 16219 target tokens. Expectedly, better coverage statistics are achieved when using less constraining methods. We also present there another figure of interest we gathered: the respective contribution of non-contiguous and contiguous units to this coverage. As can be seen, contiguous units account for most of the coverage, which means that a standard phrase-based model would probably have captured the same information. The extra coverage brought by non-contiguous units varies between roughly 10% and 20% (absolute), although it is difficult at this stage to assess how this could have translated into a better MT system.

In all cases, the coverage is very good, with, on average, roughly 75% coverage, both for the source and the target sentences.

4.3 Corrected Coverage

Raw coverage figures as we computed them only give a rough idea of the potential of TPs. The

method		source		target	
src	tgt	%-cov	%-c	%-cov	%-c
E	E	68.70	62.55	71.90	68.63
E	R	69.58	63.21	75.31	70.33
E	R+S	69.10	62.76	73.80	67.66
R	E	79.29	72.35	77.86	74.78
R	R	80.39	73.23	80.85	76.36
R	R+S	79.80	72.72	79.57	73.69

Table 3: Source and target coverage statistics. %-cov stands the percentage of tokens that are covered, and %-c indicates the percentage of tokens covered by contiguous units.

main drawback of our methodology is that *many* different overlapping units (TLs or EPs) are allowed to cover a given source or target sentence token, which might not reflect their true usefulness in a translation task, where, typically, a single translation unit is chosen to help in the translation of a given source token or groups of source tokens.

Source coverage In order to better estimate the situation, we computed a corrected coverage by applying the algorithm in Figure 5.

The idea behind this algorithm is to select the minimum number of TLs covering as much as possible of the source sentence. All 6 combinations of our match policies have been tried for this experiment. We implemented a search algorithm in a way similar to the one embedded in a translation decoder, the main difference being that we do not build a translation, but just find the decomposition of the source sentence into TLs. Therefore the score we optimise is based on the source material only.

Conceptually, the algorithm builds the set of all the *valid* hypotheses that match the source sentence S . A valid hypothesis is a set of treelets that (at least) partially covers S and satisfies a certain number of properties, the main one being that none of the dependencies captured in the set of TLs is allowed to cross another one. Once all such hypotheses are built, the algorithm picks the one with the best score. In our case it is the one which covers S the most with the minimum number of treelets.

In practice, because of the combinatorial nature of the algorithm, these hypotheses are maintained into priority queues $Stack(i)$ that sometimes have to be pruned to achieve an acceptable computation time. The i th stack contains, at most, the b best ranked valid hypothe-

```

// init
for all  $i \in [1, n]$  do
   $Stack(i) \leftarrow \phi$ 
let  $tl[i] \leftarrow \{T \in \mathcal{M} \wedge \text{s-match}(T, s_i) = \text{true}\}$ 

// the search
for all  $i \in [1, n]$  do
  for all  $T \in tl[i]$  do
     $\text{add}(\epsilon, T, 1)$ 
    for all  $j \in [1, n]$  do
      for all  $h \in Stack(j)$  do
        if  $\text{s-extend}(T, h)$  then
           $\text{add}(h, T, j + 1)$ 

// the best hypothesis
let  $best \leftarrow$  the first hypothesis
for all  $i \in [1, n]$  do
  for all  $h \in Stack(i)$  do
    if  $\text{score}(h) > \text{score}(best)$  then
      let  $best \leftarrow h$ 

```

Figure 5: Algorithm to compute the corrected source coverage. \mathcal{M} is the set of all treelets matching the source sentence S . $\text{add}(h, T, n)$ is a function which adds in $Stack(n)$ the hypothesis h extended by the treelet T . $\text{s-extend}(T, h)$ is a predicate which is true if the treelet T can extend the hypothesis h , and $\text{score}(h)$ returns the score of a hypothesis h . Please read the text for more details.

ses built of i TLs. We used $b = 500$ for our experiments. The first stack (the one with only one TL per hypothesis) is seeded with all the different treelets s-matching the source sentence, with one treelet per hypothesis. The algorithm then goes along the source positions and systematically tries to extend previously built hypotheses with all of the treelets rooted at this very source position. A treelet may extend a hypothesis only if it does not introduce dependencies that cross other ones, and if at least one dependency is added to the hypothesis.

An example of the output of this algorithm is given in Figure 6. Notice that, in this example, there were 8 candidate treelets found by s-matching , but only 4 were selected by the algorithm. Out of the 15 tokens, 8 tokens are covered (53%). Furthermore, we observe that 2 tokens are covered by the non-contiguous treelet $\{\{\text{cette@-2}\} \text{législature}\}$, which “conveniently” skips the token 33e (thirty-third in English). In this example, it also happens that “droit à la pro-

priété” (property rights) is captured here by 2 TLs, whereas it would have been captured as a single parameter in a standard phrase-based model.

This illustrates two strengths of the TP approach, at least regarding the source material and the treelets. First, a completely unknown token (33e) can be skipped by a treelet, while the tokens of the latter are still available to produce a translation for the surrounding known tokens. Second, a source token can be captured by many treelets, suggesting a way to combine them into a more elaborate tree during the decoding phase, possibly with more meaningful results.

Source sentence

Au_cours_de cette 33e législature nous avons examiné le droit à la propriété à trois égards

Treelets in corrected coverage

```

{\à@-2} {\la@-1} propriété}
{\à@-1} trois}
{\cette@-2} législature}
{\droit {\propriété@3}}
```

Figure 6: Illustration of the corrected source coverage computed by the algorithm in Figure 5. Words merged together with an underscore form a SYNTEX token.

Target coverage Once a corrected source coverage is computed, we apply another algorithm to select among all the EPs that are associated with the TLs selected, the ones that maximally cover the target sentence T , once again, with the minimum number of phrases. This algorithm is presented in Figure 7.

The candidate EPs are those associated with the TLs obtained from the corrected source coverage computation, although the algorithm would work equally well with the treelets of the raw source coverage, albeit more slowly. These candidate EPs must also t-match the target sentence. The criteria used to find the score of a coverage hypothesis are, in order of importance, the target coverage (maximisation) and the number of covering EPs (minimisation). No target token e_i is allowed to be covered by more than one EP (no overlapping EPs). However, we did allow EPs to cover the target tokens contained in the “gaps” left by another EP. For example, given the target sentence **the white rabbit**, if an EP covers the words **the** and **rabbit**, then we allow another EP to cover the

word `white` contained in the gap, if there is such an EP, naturally.

One additional constraint that this algorithm enforces is that no two EPs in the corrected target coverage can share the same source treelet in \mathcal{M} , the set of treelets matching the source sentence.

Again, to avoid a combinatorial explosion of hypotheses (stored in *HypoSet* in Figure 7), we only kept the best 10 000 hypotheses at all times.

```
// init
HypoSet ← 0-coverage hypothesis

// the search
for all T ∈ M do
  AddSet ← φ
  let ep be the set of EPs associated with T
  which t-match E
  for all p ∈ ep do
    for all h ∈ HypoSet do
      if t-extend(p, h) then
        add(p, h, AddSet)
  HypoSet ← HypoSet ∪ AddSet

// the best hypothesis
find in HypoSet the hypothesis h for which
score(h) is the highest and return it.
```

Figure 7: Algorithm to compute the corrected target coverage. \mathcal{M} is the set of all treelets matching the source sentence S . `t-extend(p, h)` is a predicate which is true if the elastic phrase p can extend the hypothesis h , `add(p, h, set)` adds to `set` the hypothesis h extended with p , and `score(h)` returns the score of a hypothesis h . Please read the text for more details.

We complete the example introduced in Figure 6 with the corresponding target coverage, presented in Figure 8. Out of the 11 target tokens, 5 are covered by 3 EPs, a 45% coverage. An interesting match has occurred: while the EP `|property@0||rights@5|` was acquired with a gap of 5 between the words `property` and `rights`, a match was possible with contiguous target words.

The corrected coverage figures are presented in Table 4. Without surprise, these figures are inferior to those reported in section 4.2, although the target coverage is the one which suffers the most from this optimisation. This may be due to the fact that the source coverage

Target sentence

This thirty-third Parliament is dealing with property rights on three different fronts

Elastic phrases in corrected coverage

```
|this@0||Parliament@1|
|property@0||rights@5|
|three@0|
```

Figure 8: Illustration of the corrected target coverage. Words merged together with an underscore form a SYNTAX token.

optimisation does not take into account the restrictions in the number of candidate EPs that it will eventually impose on the target coverage optimisation. Indeed, when we reach the target coverage optimisation, our options have been limited by the previous step.

Nonetheless, it is apparent from these results that non-contiguous units can contribute significantly to source and target coverage statistics.

method		source		target	
src	tgt	%-cov	%-c	%-cov	%-c
E	E	59.74	53.20	56.72	45.86
E	R	58.15	51.54	57.55	38.66
E	R+S	58.55	51.99	57.14	37.88
R	E	65.34	48.70	56.56	47.60
R	R	61.21	44.44	56.41	39.35
R	R+S	62.67	45.95	56.43	38.63

Table 4: Source and target corrected coverage statistics. %-cov stands for the percentage of tokens that are covered, and %-c indicates the percentage of tokens covered by contiguous units.

5 Towards EBMT

Without writing a specific decoder, it is difficult to determine whether TPs can be of help in MT. The RALI, the research group in applied computational linguistics at the Université de Montréal, is currently developing a decoder that will, hopefully, be able to handle tree-phrases. However, we could not wait for the final implementation of this decoder to measure the potential of tree-phrases in a translation context.

We therefore used *pharaoh*², a beam search decoder for phrase-based statistical machine translation models developed by (Koehn, 2004). However, since we do not have access to the code of this program, we cannot modify it to favour

²www.isi.edu/licensed-sw/pharaoh/

the treelets or phrases contained in our collection, or to propose and implement a new decoding strategy addressing our specific needs. We therefore resorted to a post-processing experiment, using a n-best list produced by pharaoh.

5.1 Experimental Set-Up

Using once again the training and test corpora described in Section 4, we had pharaoh produce a translation for the same 1000 randomly selected source sentences, as well as a n-best list of roughly 1000 different best candidates per translated sentence. We will call each source sentence S_i ($i = 1 \dots 1000$), its corresponding reference sentence R_i and its candidates $C_i[j]$ ($j > 0$). The first candidate for a sentence S_i is $C_i[1]$ and is the best one, the candidate eventually output by pharaoh as the translation of S_i .

For each of these candidates, we calculated their word error rate (wer) when compared to their respective reference translation. For each set of candidates translated from the same source sentence S_i , we called *oracle* (O_i) the candidate with the lowest wer. When multiple candidates had the same wer, we randomly selected one among the candidates tied for lowest wer.

We then proceeded to compute a variety of coverage-related features for each candidate, like we did in the previous section. We did the same for the reference target sentence R_i and for the oracle O_i . To do so, we used the exact (E) matching policies both for the source and target sentences.

Our goal was to discover, if possible, a coverage feature f for which, on average, $f(R_i) > f(C_i[1])$ or $f(O_i) > f(C_i[1])$. This would mean that our tree-phrase approach could lend itself to a translation task. Indeed, if such a feature f exists, then it means that our memory better “recognizes”, on average, R_i or O_i , than the best candidates $C_i[1]$, and the two former have the lowest word error rates: R_i has a wer of 0 by definition, and O_i is the candidate with the smallest wer. It could then be argued that our system is more likely to produce translation with lower wer’s than a typical system.

Admittedly, this is a unorthodox way of assessing the usefulness of TPs in machine translation, but this is a pilot study and the resources at hand are still limited.

We computed the features in Table 5 for all the candidates and the reference. We attempted

as well to integrate entropy-related features, but did not observe any interesting results. The results are presented in the following section.

f1	src cov. (%)
f2	trg cov. (%)
f3	src cov. w/ contiguous TLs (%)
f4	trg cov. w/ contiguous EPs (%)

Table 5: Various features computed for each candidate and reference in the n-best list for 1000 translations produced by pharaoh.

5.2 Results and Discussion

Table 6 presents the averages and standard deviations for the values of the different features introduced in Table 5, computed for the 1000 translations and their corresponding candidates. The “random” column is the average/standard deviation for each feature computed on a set composed of a randomly selected candidate for each S_i . It acts as a control group, making sure the differences we observe between the sets ref, best and oracle are not purely fortuitous.

feature	stat	ref	best	oracle	rnd
src cov	avg.	67.1	68.1	70.1	67.7
	stdev.	20.9	20.9	19.9	21.5
trg cov	avg.	70.6	71.6	75.4	70.4
	stdev.	22.0	22.1	20.4	22.2
src cov c	avg	61.0	62.4	63.7	61.8
	stdev.	21.8	21.7	21.3	22.3
trg cov c	avg.	67.9	69.4	73.4	68.4
	stdev.	22.0	22.1	20.5	22.2

Table 6: Averages and standard deviations for the values of features computed on a n-best list for 1000 translations produced by pharaoh. src cov is the source coverage, src cov c is the source coverage from contiguous units. ref is the reference R_i for each S_i , best is the first candidate $C_i[1]$, oracle is O_i , the candidate with the lowest wer, and rnd (random) is the set composed of a randomly selected candidate for each S_i . All values are expressed in percentage.

No set among ref, best, oracle clearly stands out, on average, for any of the features we chose. Nonetheless, the oracle set, the one composed of the candidates O_i with the lowest wer’s, systematically exhibits the highest scores for each feature. For the target coverage, a difference of

3.9% (absolute) is observed between the oracle and the next best contender.

This assessment strategy is farfetched, we are the first to admit it, but it may argue in favour of the treelet/elastic phrase approach at this early stage of research. If, indeed, $f(O_i) > f(C_i[1])$ like the figures in Table 6 seem to suggest, then our memory could have—at least—the potential to generate translations with lower word error rates than a classic phrase-based one, a promising perspective.

6 Discussion

We presented a pilot study aimed at appreciating the potential of Tree-Phrases as a base unit for example-based machine translation. Since we are in the early stages of this study and do not yet benefit from a decoder adapted to these units, we resorted to indirect measures of the potential of a repository populated with TPs.

Coverage statistics clearly show that, whether we allow restrictive match policies or more relaxed ones, our treelets and their corresponding elastic phrases cover most of the source and target material. We observe a slight coverage loss when we apply more rigorous match policies, but that was expected. This generally bodes well for a translation system based on TPs. We can at least rest assured that a given source sentence for which we need a translation will be recognized by the repository. Moreover, since the target coverage of the associated target sentence (reference) is also good, there is a distinct possibility that our system could generate a translation in many ways similar to the reference.

Coverage examples have also highlighted one of the most interesting features of treelets and elastic phrases: their capacity to conveniently skip unknown tokens in a given sequence of words in order to recognize the surrounding tokens. This is of major interest, since unknown or rare tokens usually confuse a standard phrase-based decoder, which does not benefit from the freedom of elastic gaps.

Our post-processing experiments using a n-best list generated by pharaoh, a phrase-based decoder, to attempt to highlight the interest of Tree-Phrases in the context of a translation met a limited success. Our somewhat unconventional approach suggests nonetheless that a TP repository could possibly generate translations with lower word error rates (compared to the reference) than those generated by a more traditional approach.

All this evidence leads us to believe that a TP-based MT system could be a viable alternative to a standard phrase-based one, that such a new repository might better generalise upon a training corpus.

Naturally, this is a preliminary study, and the metrics and features computed here as well as the conclusions drawn from them need to be validated in a more conventional approach, one that would benefit from a decoder capable of handling treelets and elastic phrases. We would then be able to directly measure the contributions of such translation units to a MT system. More efforts could also be invested in considering other translation unit pairs, namely elastic phrase-elastic phrase, or treelet-treelet.

7 Acknowledgements

This work has been financially supported by a grant from PRECARN.

References

- Didier Bourigault and Cécile Fabre. 2000. Approche linguistique pour l'analyse syntaxique de corpus. *Cahiers de Grammaire*, (25):131–151. Toulouse le Mirail.
- P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.
- Yuang Ding and Martha Palmer. 2004. Automatic learning of parallel dependency treelet pairs. In *First International Joint Conference on Natural Language Processing*.
- Daniel Gildea. 2003. Loosely tree-based alignment for machine translation. In *ACL*.
- Almut Silja Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 2005. Adaptation of the translation model for statistical machine translation based on information retrieval. In *10th EAMT*, pages 133–142, Budapest, Hungary, May 30-31.
- P. Koehn, F.J. Och, and D. 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.
- Evgeny Matusov, Stephan Kanthak, and Hermann Ney. 2005. Efficient statistical machine translation with constraint reordering. In *10th EAMT*, pages 181–188, Budapest, Hungary, May 30-31.

- F.J. Och and H. Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of ACL*, pages 440–447, Hongkong, China.
- Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 271–279, Ann Arbor, Michigan, June. Association for Computational Linguistics.
- Michel Simard and Philippe Langlais. 2003. Statistical translation alignment with compositionality constraints. In *HLT-NAACL workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 19–22, Edmonton, Canada, May.
- Michel Simard, Nicola Cancedda, Bruno Cavestro, Marc Dymetmann, Éric Gaussier, Cyril Goutte, Arne Mauser, Philippe Langlais, and Kenji Yamada. 2005. Une approche à la traduction automatique statistique par segments discontinus. In *Proceedings of the 12th conference of Traitement Automatique des Langues Naturelles (TALN)*, Dourdan, France, June 6-10.