Dynamical visualization of nested correspondences

Christophe Chenon
Groupe d'Etude pour la Traduction Automatique
CLIPS - IMAG
Grenoble
France.

christophe.chenon@imag.fr

Introduction

Computer-aided translation systems based on translation memories have spread rapidly in the translation industry over the last 15 years or so. A key reason for their success is that they require no language-specific resources. One of their main drawbacks is that they do not allow translators to reuse translation of micro-examples that are abundant in the memories. Indeed, in the classical use of translation memories, a fuzzy match will only be presented to the translator if it broadly corresponds to the segment to be translated. This granularity is far from optimal to take advantage of minor correspondences that will remain ignored by the system. These issues have been repeatedly addressed in the literature; see for example [7], [13], [20], [25], [26] and [27].

We present a new formalism, TransTree, to encode subsegmental correspondences recursively. TransTree is a by-product of a new method designed to automatically generate such correspondences, based on statistics collected from large corpora of translation memories. With this new approach to translation memories, we will be able to provide translators with a much richer material than whole bisegments, provide pedagogical examples of translations to learners, and pave the way towards construction of quasi-translated segments, to be used by translators. Because we use statistics-based methods only, we remain within the knowledge free paradigm that made possible the clear success of translation-memory-based systems of the first generation.

In this article, TransTree will first be presented on a couple of examples, along with an XML form. A VML-based graphical representation derived from the XML expression will also be given. We will then propose a dynamical, javascript-based vizualisation of corresponding subsegments in TransTree that will allow users to efficiently reuse translation of subsegments. In the second and third parts, we will show how to produce subsegmental correspondences automatically and express them in TransTree. In the conclusion, we will explain where we intend to concentrate in future works.

I The TransTree formalism

I.1 Architecture and XML representation

The architecture we propose relies on local correspondences between strings in a pair of segments. Such elementary correspondences between two strings are called prime amphigraphs. "Prime" because they cannot be further divided in simpler elements, "amphigraphs" because they contain two written strings that are considered two facets of a same entity.

Prime amphigraphs can be any pair of text strings in different languages, here English and French. We will assume that they actually connect semantically related strings, and thus bear semantical information, but that is not necessary in the formalismper se. Typically, prime amphigraphs will be of the form ("computer","ordinateur") or ("for","pour"). Prime amphigraphs with a common source or target string but with differring counterparts are different. For example ("had","avions") and ("airplanes","avions") as well as ("use","utiliser") and ("use", "utilisation") are all different prime amphigraphs. Prime amphigraphs do not contain variables or place holders: they are just pairs of strings. The XML expression we chose for prime amphigraphs is given below. It is made up of two strings, the *text* elements:

```
<amphigraph>
  <text xml:lang='en'>computer</text>
  <text xml:lang='fr'>ordinateur</text>
</amphigraph>
```

The standard *xml:lang* attribute is used to encode the language in which the content of a *text* element is written.
There is no limitation or constraint on texts. They can contain punctuation, spaces or even markup, if properly escaped. They may contain poorly delimited, non-related strings. There is no need to specify any such constraint at this point.
Then, we will consider general amphigraphs. These are similar objects, although somewhat more complex. A general amphigraph posesses not only *text* elements, but also children *amphigraph* elements. The *text* elements contain place holders, *amph* elements, that are instantiated by children amphigraphs. A mechanism to take care of nested correspondences between the two texts is provided by an occurrence indicator, the *occ* attribute, which specifies correspondences between place holders. To make things clearer, let us take an example.
We want to capture the correspondences between the English segment "dialog box" and the French segment "boîte de dialogue". Here is the amphigraph representation in XML:

```
<amphigraph>
 <text xml:lang='en'><amph occ='A'/> <amph occ='B'/></text>
 <text xml:lang='fr'><amph occ='B'/> de <amph occ='A'/></text>
 <amphigraph occ='A'>
  <text xml:lang='en'>dialog</text>
  <text xml:lang='fr'>dialogue</text>
 </amphigraph>
 <amphigraph occ='B'>
  <text xml:lang='en'>box</text>
  <text xml:lang='fr'>boîte</text>
 </amphigraph>
</amphigraph>
```

In this example there are three amphigraphs: a general amphigraph and two children prime amphigraphs. The latter bear the *occ* attribute, used to unambiguously locate

corresponding amphigraphs. The same attribute is also used in the *amph* element, and it locally plays the role of an identifier.
There are some constraints on *amph* and *occ*:

- There must be the same number of *amph* elements in both *text* elements, with the same values, in any order
- *occ* attributes being identifiers, they must have different values within a given *text* element
- there must be as many children *amphigraph* elements as there are *amph* elements in texts, one for each value of the *occ* attribute
- As a consequence, within a single amphigraph, the *occ* attribute occurs three times for one given value.

The prime amphigraphs capture the two pairs ("box","boîte") and ("dialog","dialogue"). One specificity of XML is its ability to represent both structured data and strings with inline markups. That is what allows us to capture the French preposition "de" in the general amphigraph, where it logically belongs. This flexibility of XML makes it particularly suitable to support the TransTree formalism.
This is better examplified with the following, larger example. In a translation memory, we find the following bisegment:

```
<bisegment>
 <source>This task shows you how to change views.</source>
 <target>Dans cette tâche, vous apprendrez à modifier les
vues.</target>
</bisegment>
```

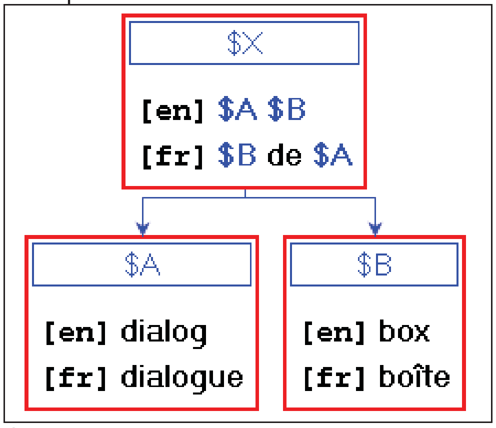
There are very heterogeneous grammatical features in this bisegment. Specifically, the first part of the English sentence is conveyed in a totally different manner into French. Here is how we can represent it:

```
<amphigraph>
 <text xml:lang='en'><amph occ='A'/> <amph occ='B'/>.</text>
 <text xml:lang='fr'><amph occ='A'/> <amph occ='B'/>.</text>
 <amphigraph occ='A'>
  <text xml:lang='en'><amph occ='A'/> <amph occ='B'/></text>
  <text xml:lang='fr'><amph occ='A'/> <amph occ='B'/></text>
  <amphigraph occ='A'>
   <text xml:lang='en'>This <amph occ='A'/> shows <amph occ='B'/> how</text>
   <text xml:lang='fr'>Dans cette <amph occ='A'/>, <amph occ='B'/>
apprendrez</text>
   <amphigraph occ='A'>
    <text xml:lang='en'>task</text>
    <text xml:lang='fr'>tâche</text>
   </amphigraph>
   <amphigraph occ='B'>
    <text xml:lang='en'>you</text>
    <text xml:lang='fr'>vous</text>
   </amphigraph>
  </amphigraph>
  <amphigraph occ='B'>
   <text xml:lang='en'>to</text>
   <text xml:lang='fr'>à</text>
  </amphigraph>
 </amphigraph>
 <amphigraph occ='B'>
  <text xml:lang='en'><amph occ='A'/> <amph occ='B'/></text>
  <text xml:lang='fr'><amph occ='A'/> les <amph occ='B'/></text>
  <amphigraph occ='A'>
   <text xml:lang='en'>change</text>
   <text xml:lang='fr'>modifier</text>
  </amphigraph>
  <amphigraph occ='B'>
   <text xml:lang='en'>views</text>
   <text xml:lang='fr'>vues</text>
  </amphigraph>
 </amphigraph>
</amphigraph>
```

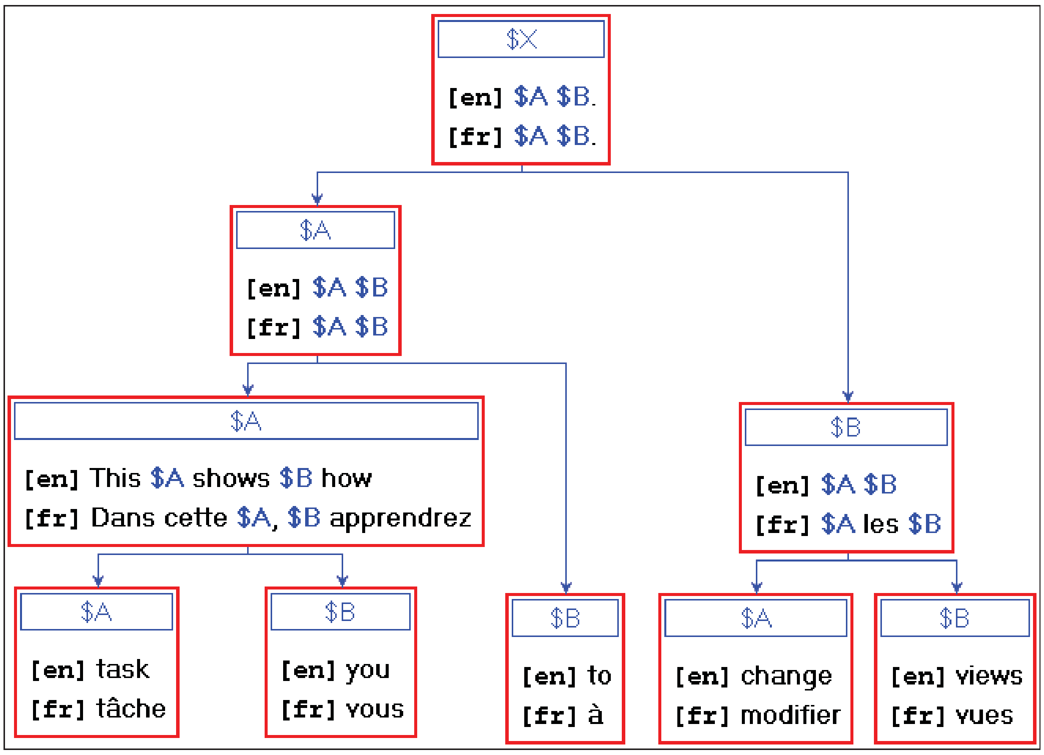
This example was built automatically, with the method described further down in this paper. The subsegments "This task shows you how to" and "Dans cette tâche, vous apprendrez à" were not broken up very much, due to the very dissimilar phrasings. This correspondence can be used as is by a translator, and would very unlikely be produced by a conventional machine translation system. Note also that the comma has been seamlessly taken into account.
As can be seen, we now have a tree that describes both segments simultaneously. The structures of both segments are determined by their counterparts. An amphigraph can contain many nested amphigraphs. The leaves of the tree are the prime amphigraphs which play a central role in this construction, as will be explained later.

I.2 Graphical visualization of the arborescence

An XSL stylesheet produces a Vector Markup Language (VML) visualization on the fly from the XML representation. Here is the graphical visualization of the first example above.



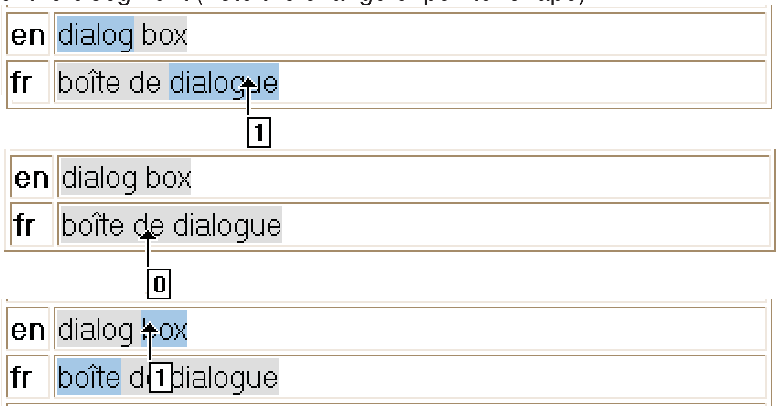
On this picture, occurrence indicators are represented with a $ sign.
Here is the same visualization of the larger example:



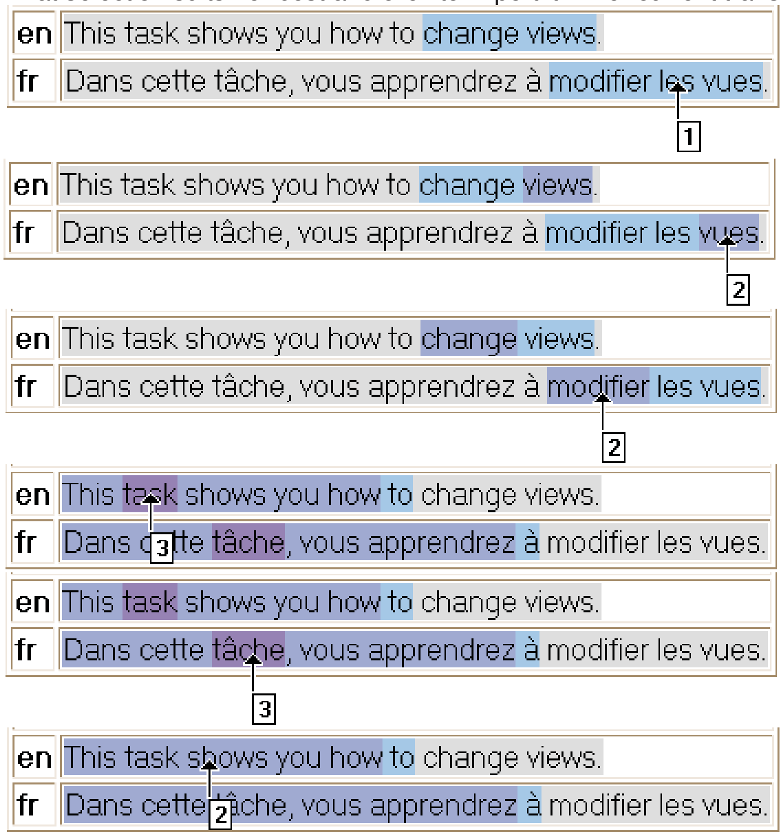
This example demonstrates how several levels of correspondences can be encoded in TransTree. There is no specific order in the branches of the tree since the linear order of each segment is supported by occurrence indicators.

I.3 Dynamical visualization of nested correspondences

Segments are more legible when displayed on the same line... The potential user of this recursive segmentation representation probably wants to read segments in a user-friendly manner. In order to help grasp the different segmentation levels while keeping some sense of ergonomy, we propose a dynamical visualisation that could be more relevant to a full-fledged computer-aided translation environment.
The main idea is that hovering with the pointer on a particular part of either segment should highlight the hierarchy of nested subsegments in both segments simultaneously. We use several colors, contrasting in darkness, to this purpose. On the next picture, one can see what happens when the user hovers on different parts of the bisegment (note the change of pointer shape).



On each image, different parts of the bisegment are highlighted, with a indication of the depth of the correspondence. The deeper correspondences highlight shorter chunks. When changing words, a different branch of the tree is selected.
The second example involves a much larger tree. At any time, the user can decide what selection suits her best and click to import it in her current translation stream.



en	This task shows you how to change views.
fr	Dans cette tâche, vous apprendrez à modifier les vues.

en	This task shows you how to change views.
fr	Dans cette tâche, vous apprendrez à modifier les vues.

en	This task shows you how to change views.
fr	Dans cette tâche, vous apprendrez à modifier les vues.

en	This task shows you how to change views.
fr	Dans cette tâche, vous apprendrez à modifier les vues.

en	This task shows you how to change views.
fr	Dans cette tâche, vous apprendrez à modifier les vues.

en	This task shows you how to change views.
fr	Dans cette tâche, vous apprendrez à modifier les vues.

This dynamical vizualization seems equally suitable for other types of users, such as students of a foreign language.

Conclusion of part I

The TransTree formalism that we propose captures nested correspondences in a bisegment. It can be extended to allow for other types of information, such as linguistic characterization of units or monolingual dependences although that is not directly in the scope of this article. It can also encode correspondences in more than two languages.
The XML expression of TransTree leads to several kinds of representations, that can be used by both machines and human users, such as linguists, translators or students.
The next part will show how it is possible to build TransTree structures from statistical computation on a big corpus.

II Segmentation algorithm

Naturally, one would like to use this formalism to store relevant knowledge. In a computer aided translation environment, acquiring this data manually is clearly out of the question. Also, compliance with the language-independent translation memory paradigm requires not using potential linguistic knowledge.
For these reasons, we have developed statistical heuristics to produce reliable segmentation and correspondences. We use a translation memory or, rather, a large collection of translation memories as a bilingual corpus to train the system. In

translation memories, segments are aligned during the process of acquisition/translation. Thus, segment alignment is not a concern to us.

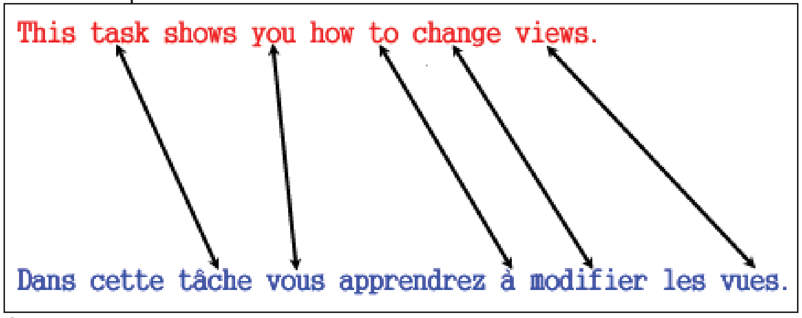
II.1. Prime amphigraphs acquisition

The first idea is to collect word correspondences throughout the corpus. Words, for us, are space-delimited strings. Other delimiters are also used, such as start and end of segment, punctuation etc. The immediate consequence of this is that we do not attempt to work on scripts without such delimiters. That bars us from tackling Chinese, Japanese, Korean or Thai, to name just a few.

To create prime amphigraphs, we establish correspondences between words. We will have to set some conditions: we want correspondences to be non-directional and injective, i.e. words in either segment have at most one counterpart.

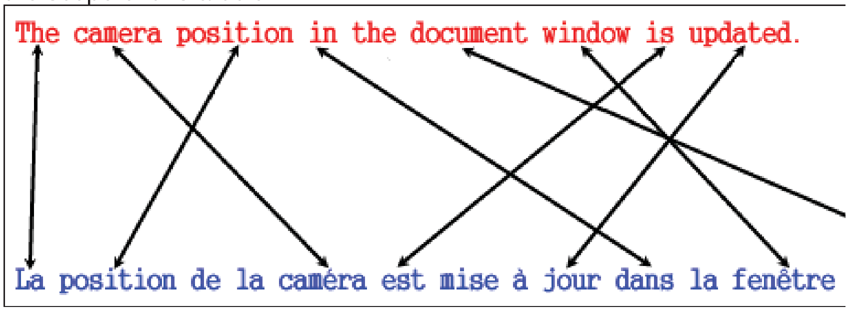
One of the simplest methods to get word correspondences consists in comparing mutual information, or some other statistical index characterizing correlation, between source words and target words as computed throughout the whole corpus. Then some algorithm can be applied such as the following one. For each word in a given segment, one can order words in the opposite segment according to mutual information. So, for any word in the first segment, some matching priority is given to words of the second segment and vice-versa. Every pair of words that grant each other first priority are in correspondence and are taken out of the bisegment. The set of remaining words is then considered in turn and so on until some condition is reached, for example when one segment is empty. For further details about these techniques, see [18].

Here are two examples of such alignments. The following two pictures are SVG graphics, produced from the XML representation:


This task shows you how to change views.
Dans cette tâche vous apprendrez à modifier les vues.

Some problems may arise when there are identical words in the segments or when mutual information leads to ties. There is a need to make a choice if we want to have actual links between occurrences of words within the two segments. In the examples shown here, these links have been omitted: the English word "the" (lower case) is not aligned on either French word "la" (lower case). The second "la" would be the correct alignement.

Methods to compute these correspondences are beyond the scope of this article.


The camera position in the document window is updated.
La position de la caméra est mise à jour dans la fenêtre

As can be seen, at the end of such a process, some words are in correspondence with the words of the opposite segment, some are not. Pairs of words in correspondence make up prime amphigraphs. Words are just strings; prime amphigraphs are largely disambiguated semantically and syntactically by the relationship that is established between their components. In the case of a single word term in either language corresponding to a multiple word term, only one link will be made. In the case of the English word "updated" and its French translation "mise à jour", a single link is established, between "updated" and "jour". The word "jour", because of its correspondence with "updated", conveys some extra information. The French words "mise" and "à" are left alone. Such "simple words" also acquire extra information: that of having no counterparts. We will use amphigraphs here as a way to manipulate more specific objects than simple words wherever possible. This will make the following process even more accurate.

II.2. Binary trees and secability

The next step consists in counting n-grams in the corpus, with "n" being typically 1, 2 and 3 and "grams" being both simple words (i.e. without correspondence link) or prime amphigraphs.

This material will enable us to compute a "secability" index on each delimiter of the segment. To make things simple, let's assume that separators are white spaces only. On each separator in a segment, we can compute an estimation of the mutual information of the two parts of the segment thus delimited by estimating its value on a small window, for example two words before and two words after the separator. Thus, we have a score on each separator on the segment. We can then assign a rank on separators, from 1 where the mutual information is the highest, to n-1 (for n words) where the mutual information is the lowest. We call this rank secability. It gives us an indication on whether delimitating chunks on this separator is more or less appropriate. The fact that we only consider ranks, i.e. whole numbers, and not mutual information values, should be emphasized. In the

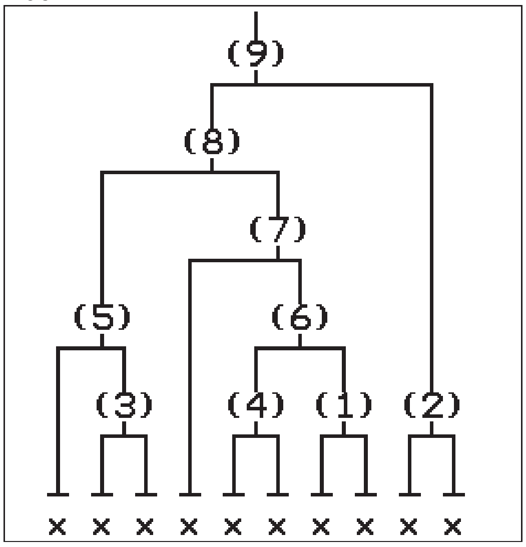
case of ties, the leftmost separator is arbitrarily given a higher secability index.
For instance, on this schema, where 'x's stand for different tokens in the segment, secability indexes are placed under each white space separator.

```
x x x x x x x x x x
 5 3 8 7 4 6 1 9 2
```

Computing secability indexes on a segment is the same as bracketing the segment. We split the segment in two parts at the highest secability point. Then we split each subsegment in two parts, at the locally highest secability point. This is the final result, from the example above:

```
(((x (x x)) (x ((x x) (x x)))) (x x))
    5  3    8  7   4  6   1      9  2
```

It is also equivalent to defining a binary tree. On this picture, each node of the binary tree bears a secability index.

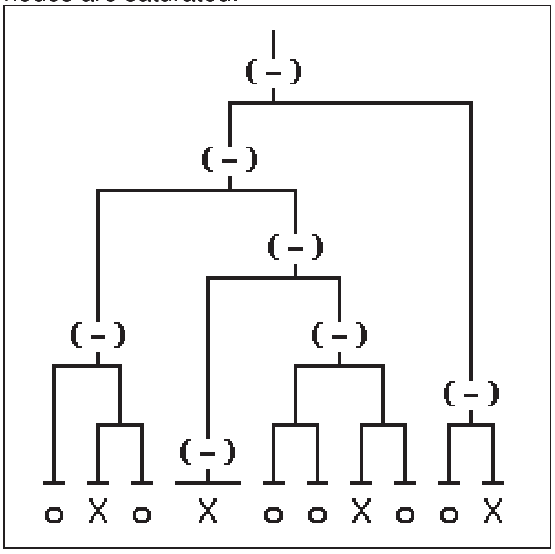


In what follows, we will refer to high nodes, which are closer to the root, and low nodes which are closer to the leaves. Thus, the root of the tree is the top node while leaves are at the bottom (the metaphor follows the drawings more than Mother Nature). The leaves of the tree are the segment tokens: prime amphigraphs and words. We will say that a node "dominates" other nodes or leaves when it is above them, i.e. closer to the root.

II.3. Alignment of subsegments

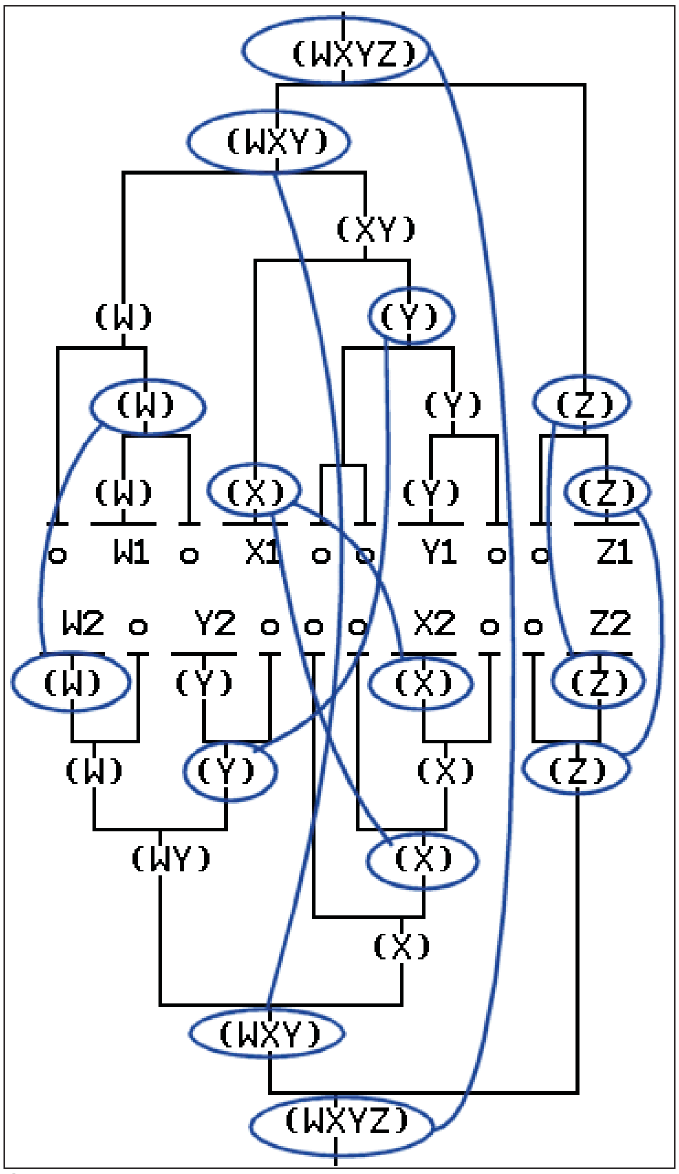
To convert pairs of binary trees obtained in the two segments of a bisegment into a single abstract tree of amphigraphs, a TransTree structure, we will study several options to map them on one another. We first exhibit two properties of nodes in binary trees, based on prime amphigraphs: "saturation" and "congruence". These properties concern the geometry of the binary trees.

a- Saturation
We will call "saturation" a property of some particular nodes in a binary tree. A node that dominates a given set of prime amphigraphs, such that any node above it dominates a strict superset of prime amphigraphs will be called a saturated node. In the figure below, saturated nodes are marked with a hyphen **(—)**. Note that not all nodes are saturated:



In this figure, 'X's stand for prime amphigraphs and 'o's for simple words. Horizontal bars are placed at saturated nodes. A given saturated node dominates either two saturated nodes or two non saturated nodes. The root node is always considered saturated, and a leaf can be saturated or not. It is possible for a saturated node to be dominated by a non saturated node (not on this picture).

b- Congruence
Having built binary trees on both segments of a bisegment, a new property emerges: some pairs of nodes in the two trees dominate the same set of prime amphigraphs. We call that property congruence. Pairs of congruent nodes ignore simple words that they dominate.



On this picture, two binary trees have been represented, one for each segment of the bisegment. The bisegment has four prime amphigraphs, labeled W,X,Y and Z. They link words 'W1' through 'Z2' in languages L1 and L2. Each node of the binary trees dominating at least one prime amphigrah is decorated with the set of one or more prime amphigraph(s) that it dominates. Because there are many pairs of congruent nodes, we have materialized only a few for the sake of clarity. The reader is invited to find them all...

With our definition, a given set of prime amphigraphs can be involved in many pairs of congruent nodes. The simplest congruent nodes are prime amphigraphs: we

have materialized an example of a prime amphigraph on this sketch, between the two (X) leaves.
An important feature appears on this drawing: there can not be congruent nodes involving sets {W,X}, {X,Y} or {W,Y}. The only possibility is a pair of congruent nodes for the set {W,X,Y}. Thus, the congruence property provides us with indications as how to determine an optimal granularity in the analysis of the segment, based on both monolingual and bilingual constraints.

c- Alignment

Basically, any pair of congruent nodes can be rewritten as a general amphigraph, without considering further structuration:
Thus, the pair of congruent nodes involving {W,X,Y}, shown on the sketch above, could be converted in the following general amphigraph:

```
<amphigraph>
 <text xml:lang='L1'>o <amph occ='A'/> o <amph
occ='B'/> o o <amph occ='C'/> o</text>
 <text xml:lang='L2'><amph occ='A'/> o <amph
occ='C'/> o o o <amph occ='B'/> o</text>
 <amphigraph occ='A'>
  <text xml:lang='L1'>W1</text>
  <text xml:lang='L2'>W2</text>
 </amphigraph>
 <amphigraph occ='B'>
  <text xml:lang='L1'>X1</text>
  <text xml:lang='L2'>X2</text>
 </amphigraph>
 <amphigraph occ='C'>
  <text xml:lang='L1'>Y1</text>
  <text xml:lang='L2'>Y2</text>
 </amphigraph>
</amphigraph>
```

This is the fundamental method to create amphigraphs, based on the comparison of two binary trees.

Conclusion of part II.

Prime amphigraph acquisition and binary trees are the basic blocks of a statistics-based TransTree construction. Several strategies to convert pairs of binary trees into the TransTree formalism are possible, that are beyond the scope of this article. We will now examine how we can use the TransTree formalism in a real computed-aided translation scenario.

III. A possible application of TransTree

How can we use the TransTree formalism to help a human translator get the most of an automatically-computed fuzzy match, in a translation-memory-based translation environment?

III.1 A simple scenario

We suppose that we have a segment to translate, and that the system selected a fuzzy match from a translation memory. A fuzzy match is a source segment for which there is a known translation that almost matches the source segment to be translated. In a translation environment such as Translation Manager, only the target side of the fuzzy match is shown. To display the original source segment of the fuzzy match, an additional action has to be taken. The available information is then the following:

1. Segment to be translated, in the source language,
2. Target language version of the fuzzy match,
3. Optionally, source language version of the fuzzy match.

Identifying useable parts of the fuzzy match can be rather confusing with only 1. and 2. because of the difference of languages. Requesting the display of 3. helps the translator to understand why the fuzzy match is indeed considered a fuzzy match by the system but also adds supplementary information that may cause confusion. Further adding to the confusion, the system often proposes more than one fuzzy match.

Let's see how the use of TransTree can help the user grasp the useable information faster.

III.2 Determining useable information in a glimpse

The segment to be translated and its fuzzy match can be compared to determine where the two strings are identical and where the differences lie, in the source language.

For example we have a pair like this:

<segment>You must choose a variable within the specified block.</segment>

<fuzzy>You must click on the variable within its scoping block.</fuzzy>

We separate common parts from differing parts by parenthesizing out substrings that do not belong to both strings, thus:

<segment>You must (choose a) variable within (the specified) block.</segment>

<fuzzy>You must (click on the) variable within (its scoping) block.</fuzzy>

We know now where the differences lay between the two segments in the source language, but without correspondences, we cannot use this material to help the translator any further. The TransTree formalism will give us some clues.

TransTree gives a hierarchical view of correspondences of different lengths between the two segments of a bisegment. Although individual words may not have a directly-matching words, correspondences can be found between larger chunks of text. We now need to determine which corresponding parts of the fuzzy match are relevant to the translator.

A more reader-friendly, static way of expressing nested correspondences between the source and target segments consists in bracketing chunks in both segments. Numbers in superscript identify prime amphigraphs, while lower-case letters in subscript denote general amphigraphs.

[[[[You]^1 [must]^2]$_a$ click [on]^3]$_b$ [[the]^4 [[variable]^5 within [its]^6 [scoping]^7 [block]^8.]$_c$]$_d$]$_e$

[[[[Vous]^1 [devez]^2]$_a$ cliquer [sur]^3]$_b$ [[la]^4 [[variable]^5 dans [son]^6 [bloc]^8 [englobant]^7.]$_c$]$_d$]$_e$

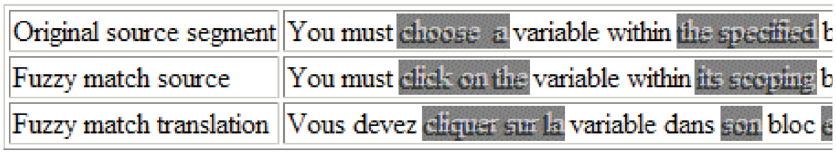
The algorithm we use to match the-parenthesized English string and the bracketed English string works as follow: we can use a general amphigraph whenever all its content is not in a parenthesis, i.e. when it is common to both the fuzzy match and the segment to be translated. The bracketed substrings that appear at a lower level are not taken into account. Thus, we can make use of the following bracketed substrings, indexed **a** and **c,** from the English string:

[[You]^1 [must]^2]$_a$

[[variable]^5 within [its]^6 [scoping]^7 [block]^8.]$_c$

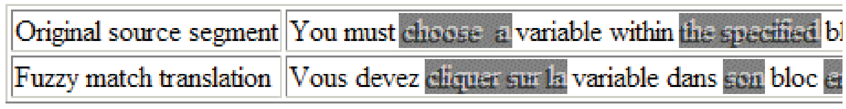
This rule enables us to transpose parts of the fuzzy match for which we have no prime amphigraph equivalent, here the English word “within”.

In the translator environment, this gives way to an implementation where only relevant parts of the fuzzy mach are active. If we compare the three segments in play, we get the following picture, where non-active parts of the segments have been disabled:



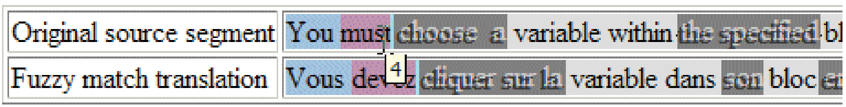
Original source segment	You must choose a variable within the specified b
Fuzzy match source	You must click on the variable within its scoping b
Fuzzy match translation	Vous devez cliquer sur la variable dans son bloc e

Even without the help of the source segment of the fuzzy match, usable parts are easy to spot out, thus:



Original source segment	You must choose a variable within the specified bl
Fuzzy match translation	Vous devez cliquer sur la variable dans son bloc e

The translator can then choose either to use this material in her translation or to use parts of it. At any moment, she can check what part of the original source segment matches what part of the translated fuzzy match. In the example given below, matching parts keep their dynamical behavior, so that the translator can wander with the pointing device and highlight matching chunks.



Original source segment	You must choose a variable within the specified bl
Fuzzy match translation	Vous devez cliquer sur la variable dans son bloc en

If the translator decides to import all or parts of the relevant chunks of the fuzzy match translation into her editing area, the system can keep a trace of the parts from the original segment that are now translated. If another fuzzy match is available, missing chunks may be translated in turn.

Conclusion of part III.

TransTree is a formalism suitable to matching chunks during the translation process. This can be used to help translators identify the relevant parts of a system-selected, translated fuzzy match automatically in a memory-based translation environment. Moreover, supervised translation by chunks can be performed with accuracy and speed.

Conclusion and further work

In this article, we have described a new formalism to represent nested subsegmental correspondences in a bisegment as an abstract tree, that we call TransTree. A dynamic presentation of data encoded with this formalism can be used to help translators benefit from pre-translated material of smaller size than whole segments. It is also suitable for other types of users, such as students of foreign languages. TransTree can easily be extended to support linguistic data and to encode correspondences between three or more languages.

We have also given a general methodology to programmatically compute a TransTree representation of a given bisegment that relies solely on aligned bilingual material such as can be found in large translation memories. Several implementations can be derived from this methodology.

We have then given a simple example of how TransTree can be used in a translation-memory-based computer-aided translation environment, leading to accurate and flexible supervised translation by chunks.

Bibliography:

[1] (1994) A. Ratnaparkhi, S. Roukos, and R. T. Ward *A Maximum Entropy Model For Parsing.* in International Conference on Spoken Language Processing, Conference Proceedings, Yokohama, pp: 803-806

[2] (2003) Al-Adhaileh, Mosleh Hmond *Synchronous Structured String-Tree Correspondence (S-SSTC) and its applications for machine translation.*, Thesis, Ph. D., UTMK, Universiti Sains Malaysia

[3] (1998) Al-Adhaileh M. H., Tang E. K. *A Flexible Example-Based Parser Based on the SSTC* in 17th International Conference on Computational Linguistics (COLING'98), Conference Proceedings, Montreal, Canada, pp: 687-693

[4] (1999) Al-Adhaileh M. H., Tang E. K. *Example-Based Machine Translation Based on the Synchronous SSTC Annotation Schema* in Machine Translation Summit VII '99, Conference Proceedings, Singapore

[5] (2002) Al-Adhaileh M. H., Tang E. K. *Synchronous Structured String-Tree Correspondence (S-SSTC)* in IASTED02, Conference Proceedings, Innsbruck, Austria

[6] (2003) Bowen Zhou, Y. Gao, J. Sorensen, D. Dechel and M. Picheny *A Hand-held Speech-to-speech Translation System* in IEEE ASRU, Conference Proceedings, US Virgin Islands
[7] (2000) Brown, Ralph D. *Automated Generalization of Translation Examples* in COLING 2000, Conference Proceedings, vol: Volume 1
[8] (1997) Charniak., E *Statistical Techniques for Natural Language Parsing* in AI Magazine, Magazine Article, vol: 18, pp: 33-35
[9] (2002) Dario Benedetto, Emanuele Caglioti, Vittorio Loreto *Language Tree and Zipping* in Physical Review Letters, Magazine Article, vol: 88
[10] (1990) David M. Magerman, Mitchell P. Marcus *Parsing a Natural Language Using Mutual Information Statistics* in National Conference on Artificial Intelligence, Conference Proceedings
[11] (2000) Foster, George and Langlais, Philippe *Using Context-Dependent Interpolation to Combine Statistical Language and Translation Models for Interactive Machine Translation* in Content-Based Multimedia Information Access (RIAO), Conference Proceedings, Paris, France, pp: 507-518
[12] (2001) Franz Josef Och, Hermann Ney. *Statistical Multi-Source Translation* in MT Summit 2001, Conference Proceedings, Santiago de Compostela, Spain, pp: 253-258
[13] (2000) Furuse, Osamu and Planas, Emmanuel *Multi-level Similar Segment Matching Algorithm for Translation Memories and Example-Based Machine Translation* in Coling 2000, Conference Proceedings
[14] (1998) Gaussier, Eric *Flow Network Models for Word Alignment and Terminology Extraction from Bilingual Corpora* in ACL 1998, Conference Proceedings, pp: 444 -- 450
[15] (1996) Goblirsch, D. M. *Viterbi Beam Search with Layered Bigrams* in ICSLP '96, Conference Proceedings, pp: 2131-2134
[16] (2001) Goodman, Joshua T. *A Bit of Progress in Language Modeling Extended Version*, Report, Technical Report, Microsoft
[17] (2003) Gow, Francie *Metrics for Evaluating Translation Memory Software*, Thesis, MA Thesis, School of Translation and Interpretation, University of Ottawa
[18] (2001) Kraif, Olivier *Constitution et exploitation de bi-textes pour l'aide à la Traduction*, Thesis, Thèse de doctorat, Université de Nice Sophia Antipolis
[19] (1994) Kumano, A. and Hirakawa, H., *Building an MT Dictionary from Parallel Texts Based on Linguistic and Statistical Information* in COLINC, Conference Proceedings, pp: 76-81
[20] (2001) Langlais, Philippe and Simard, Michel *Sub-sentential Exploitation of Translation Memories* in

Machine Translation Summit VIII, Conference Proceedings, Santiago de Compostela
[21] (1997) Melamed, I. Dan *A Word-to-Word Model of Translational Equivalence* in Association for Computational Linguistics (ACL'97), Conference Proceedings, Madrid, Spain
[22] (2000) Ney, H. and Och, F. *A Comparison of Alignment Models for Statistical Machine Translation* in Coling 2000, Conference Proceedings, Saarbrucken, Germany
[23] (1992) P.F. Brown, V.J. Della Pietra, P.V. deSouza, J.C. Lai and R.L. Mercer *Class-based n-gram models of natural language.* in Comp. Linguistics, Book Section, vol: 18(4), pp: 467-479
[24] (2003) Philippe Langlais, Michel Simard *De la traduction probabiliste aux mémoires de traduction (ou l'inverse)* in TALN 2003, Conference Proceedings, Batz-sur-Mer
[25] (1999) Planas, E. & Furuse O *Formalizing Translation Memories* in Machine Translation Summit VII, Conference Proceedings, Singapore, pp: 331-339
[26] (2000) Planas, Emmanuel *Extending Translation Memories*, Report, NTT Cyber Solutions Laboratories, Japan
[27] (2000) Planas, Emmanuel *A Case Study on Memory Based Machine Translation Tools*, Report
[28] (1995) S. Martin, J. Liermann, H. Ney *Algorithms for Bigram and Trigram Word Clustering.* in EUROSPEECH-95, Conference Proceedings, Madrid, pp: 1253-1256
[29] (2001) Schäler, Reinhard *Beyond Translation Memories* in MT Summit VIII, Conference Proceedings, Spain
[30] (1996) Su Keh-Yih, Chiang Tung-Hui and Chang Jing-Shin *An Overview of Corpus-Based Statistics-Oriented (CBSO) Techniques for Natural Language Processing* in Intl. Journal of Computational Linguistics and Chinese Language Processing (CLCLP), Book Section, Taipei, vol: 1(1), pp: 101-157