The AlignmentSet Toolkit
Lambert, P.
Research Report LSI-04-37-R
The Alignment Set Toolkit

Patrik Lambert  (lambert@lsi.upc.es)

TALP Research Center
Universitat Politècnica de Catalunya
Jordi Girona Salgado, 1-3, 08034 Barcelona, Spain
lambert@talp.upc.es

Version 1.0

Abstract
The Lingua-AlignmentSet distribution is a Perl Tools Library (and command-line utilities) to handle an Alignment Set, i.e. a set of sentences aligned at the word (or phrase) level. It provides methods to display the links, to apply a function to each alignment of the set, to evaluate the alignments against a reference, and more. One of the objectives of the module is to allow the user to perform all these operations without bothering with the particular physical format of the Alignment Set. Anyway it also provides format conversion methods.
Contents

1 Description 3
  1.1 The Alignment Set ........................................... 3
  1.2 Description of storage formats ............................ 3
  1.3 Visualisation representations and formats ............... 5
  1.4 Evaluation .................................................... 6
     1.4.1 Unlinked words representation ...................... 7
     1.4.2 Link weights ........................................... 8
     1.4.3 Implementation ....................................... 8
  1.5 Symmetrisation .............................................. 8

2 Reference 9
  2.1 Alignment Set definition .................................... 9
     new function .................................................. 9
  2.2 Conversion between formats ................................ 10
     chFormat method ............................................ 10
  2.3 Visualisation .................................................. 11
     visualise method ............................................ 11
  2.4 Evaluation ..................................................... 12
     evaluate method ............................................ 12
     The AlignmentEval class .................................. 12
  2.5 Alignment processing ........................................ 13
     processAlignment method .................................. 13
     symmetrize method ......................................... 14
  2.6 Miscellaneous ................................................ 15
     chooseSubsets method ..................................... 15

3 Known problems 15

4 To Do List 15

5 Acknowledgements 16
1 Description

1.1 The Alignment Set

An Alignment Set is a set of pairs of sentences aligned at the word (or phrase) level.

We refer to file set as the files containing the alignment information in a given format (only one file in Giza format, three separate files in Naacl format—cf section 1.2). In theory an Alignment Set could be physically contained in one or more file sets, and could be only a part of each file set. In the current implementation, it can only be contained in one file set. However, it can be simply a part of this file set.

The attributes of an Alignment Set are (for exact details see the reference part, section 2.1):

file sets The array of all the file sets where the Alignment Set is physically stored. In the current implementation this array can only contain one element. The attributes of a file set are:

location A hash containing the paths of all the files and directories where the alignment information is stored. In some formats each component of the file set is stored in a separate file. The way the location hash has to be specified for each format is explained in section 2.1.

format The format of the file(s) where the Alignment Set is stored (cf section 1.2).

range The first and last sentence pairs to be included in the Alignment Set.

In this toolkit an Alignment Set is a perl object, whose reference is passed to the methods that use or process this Alignment Set.

1.2 Description of storage formats

In this section we describe the most widely used formats to store Alignment Sets in files.

GIZA The format used by the Giza toolkit (Al-Onaizan et al., 1999; Och, 2000). All information is contained in only one file, but since the alignment produced by the toolkit is not symmetric, up to two files can be specified: source-to-target alignment file and target-to-source alignment file. The alignment of each sentence pair in GIZA occupies three lines of the file:

# Sentence pair (1) source length 15 target length 17 alignment score : 9.53025e-19
es que el dieciocho , francamente es del todo imposible , no puedo encontrar .

NULL ( { 13 } ) it's ( { 1 } ) that ( { 2 } ) the ( { 3 } ) eighteenth ( { 4 5 } ) ,
( { 6 } ) frankly ( { 7 } ) that's ( { 8 } ) totally ( { 9 10 } ) impossible ( { 11 } ) ,
( { 12 } ) i ( { 14 } ) can’t ( { 15 } ) find ( { 16 } ) anything ( { } ) . ( { 17 } )

The same sentence with source and target reversed could be:

# Sentence pair (1) source length 17 target length 15 alignment score : 1.12222e-22
it's that the eighteenth , frankly that's totally impossible , i can’t find anything .

NULL ( { } ) es ( { 1 } ) que ( { 2 } ) el ( { 3 } ) día ( { } ) dieciocho ( { 4 } ) ,
( { 5 } ) frankly ( { 6 7 8 } ) es ( { } ) del ( { } ) todo ( { } ) impossible ( { 9 12 } ) ,
( { 10 } ) no ( { } ) le ( { } ) puedo ( { 11 } ) encontrar ( { 13 14 } ) .
( { 15 } )

NAACL This format is described in the summary paper of the HLT-NAACL 2003 Workshop on Building and Using Parallel Texts (Mihalcea and Pedersen, 2003).

The Alignment Set is stored in three separate files: one for the source sentences, one for the target sentences and one for the alignments.
source sentences file One line per sentence. The sentence number is marked as well as the end of the sentence:
<s snum=0008> hear, hear! </s>
<s snum=0009> Mr. Speaker, my question is directed to the Minister of Transport. </s>

target sentences file The format is the same as for the other sentences file:
<s snum=0008> bravo! </s>
<s snum=0009> monsieur le Orateur, ma question se adresse à le ministre chargé de les transports. </s>

alignments file There is one line per link. First the sentence number is indicated, then the number of the source token, then the number of the corresponding target token. Two optional marks are S(sure)/P(possible) and the confidence in the link (not present in this example):

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0008</td>
<td>4</td>
<td>2</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>1</td>
<td>1</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>2</td>
<td>1</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td>3</td>
<td>1</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>1</td>
<td>1</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>2</td>
<td>3</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>3</td>
<td>4</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>4</td>
<td>5</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>5</td>
<td>6</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>8</td>
<td>9</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>9</td>
<td>10</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>10</td>
<td>11</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>11</td>
<td>13</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>12</td>
<td>15</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>13</td>
<td>16</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>2</td>
<td>2</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>6</td>
<td>7</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>6</td>
<td>8</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>7</td>
<td>7</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>7</td>
<td>8</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>11</td>
<td>14</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>12</td>
<td>14</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0009</td>
<td>0</td>
<td>12</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BLINKER This format has been introduced in the Blinker project (Melamed, 1998a) and is used by the Alpaco alignment editor (Pedersen and Rassier, 2003). This strict syntax has been conceived for the manual annotation of a corpus of sentence pairs. There is one file for the source sentences, one for the target sentences and a directory for each annotator, containing one alignment file for each sentence pair. The two sentences files have the syntax EN.sample.<snb> and FR.sample.<snb> where <snb> is the sample number (minuscule and capital letters must be respected). The annotator’s directories are named A<snb> (<snb> is the annotator number) and are situated in the same directory as the sentence files. The alignment files, in each annotator’s directory, must be named samp<snb>.SentPair<pnb>, where <pnb> is the sentence pair number. In the alignment file there is one line per link, containing the source token number and the corresponding target token number.

NOTE: in this library, only the alignment file name syntax (samp<snb>.SentPair<pnb>) must be strictly respected (see section 2.1).

The best way to describe it is with an example:

directory tree A typical directory tree of a Blinker Alignment Set would be:
(project name)/

A1/
samp1.SentPair0
samp1.SentPair1
samp1.SentPair2

A2/
samp1.SentPair0
samp1.SentPair1
samp1.SentPair2

EN.sample.1
FR.sample.1

FR.sample.<snb> file It contains one line per sentence.
no , yo estaba pensando más hacia el seis , siete .
de acuerdo , déjame que mire .

EN.sample.<snb> file Same as the FR.sample.<snb> file.
no it isn't , i was thinking more for about the sixth or the seventh .
right , let me take a look .

samp<snb>.SentPair<snb> file The file corresponding to the first sentence pair of the
previous example could be:
1 1
2 1
3 1
4 2
5 3
6 4
7 5
8 6
10 7
11 8
12 9
13 10
15 11
14 11
16 12
9 0

1.3 Visualisation representations and formats

The most popular ways of representing visually an alignment between two sentences are
drawing lines between linked words (figure 3) or marking the intersection of two linked words
in a matrix (fig. 2). Another possibility is to simply enumerate the links (fig 1). Anyway
there is always the possibility to convert the Alignment Set files to BLinker format and visualise
them using the Alpaco editor (see section 1.2).

The visualise function outputs a file in one of these representations:

enumLinks This representation is available in two formats:

text The output is a text file that contains undirected correspondences for each sentence
pair, displayed as a succession of lines of the form: (source word<→aligned target
words).

latex Writes a (LaTeX) file where each sentence pair is represented as in figure 1: ev-
ey link is enumerated, and links are directed (source-to-target, target-to-source or
both). The alignment is shown in a (source token ← target token) form where "→" corresponds to "→","←" corresponds to "←" and "→" corresponds to "→" in the
matrix representation with “cross” marks (see below).
matrix Write a file with, for each sentence pair, its number, the two sentences and a (source × target) matrix representation of the alignment, as in figure 2. The first column contains the source tokens and the last row the target tokens. Two parameters determine the maximum number of rows and columns that can be displayed in one matrix. If the number of columns (target words of the alignment) is greater than the maximum, the matrix is split in various matrices (each matrix having all rows). If the number of rows is greater than the maximum, the alignment is displayed in the enumLinks representation. The available formats so far are:

latex Note that some features of the graphics package used in the files cannot be displayed by the dvi viewers, the solution is to create .ps files and view them with a postscript viewer.

Depending of the type of information you want to observe, you may prefer a character or another to mark the links:

cross This mark is appropriate to highlight the non-symmetry of alignments. In each row (corresponding to a source token) an horizontal dash "-" is written in each column corresponding to a target token that is aligned with it. Reversely, in each column (corresponding to a target token) a vertical dash "|" is written in each row corresponding to a source token aligned with it. A matrix point with both "|" and "-" is marked as "+". GIZA note: if a file comes from a Giza training, there can't be more than one vertical dash in each line or more than one horizontal dash in each column (the dashes tend to be oriented parallel to the lines and not perpendicular to them).

ambiguity Sure links are marked with "S" and possible (ie ambiguous) links are marked with "P". If a link has no ambiguity information, cross marks are used instead.

confidence Links are marked with their confidence (or probability) between 0 and 1. If a link has no confidence information, cross marks are used instead.

personalised mark Links are marked with the mark you pass as argument (don't forget it will be inserted in a Latex file).

NOTE: if there exists a targetToSource alignment, its links will be marked with the same type of mark as the sourceToTarget alignment, but rotated from 90 degrees to the left.

drawLines (Not implemented yet) The idea is to produce a file with, for each sentence pair, its number, the two sentences and a picture with the tokens aligned horizontally or vertically, with lines drawn between linked tokens.

1.4 Evaluation

A consensus on word alignment evaluation methods has started to appear. These methods are described in (Mihalcea and Pedersen, 2003). Submitted alignments are compared to a
NULL ¿ cuántas personas van?
NULL how many people are travelling?

Figure 2: matrix representation, \texttt{latex} format, cross mark

Figure 3: drawlimes representation

manually aligned reference corpus (gold standard) and scored with respect to precision, recall, F-measure and Alignment Error Rate (AER). An inherent problem of the evaluation is the ambiguity of the manual alignment task. The annotation criteria depend on each annotator. Therefore, (Och and Ney, 2003) introduced a reference corpus with explicit ambiguous (called P or Possible) links and unambiguous (called S or Sure) links. Given an alignment $\mathcal{A}$, and a gold standard alignment $\mathcal{G}$, we can define sets $A_S$, $A_P$ and $G_S$, $G_P$, corresponding to the sets of Sure and Possible links of each alignment. The set of Possible links is also the union of S and P links, or equivalently $A_S \subseteq A_P$ and $G_S \subseteq G_P$. The following measures are defined (where $T$ is the alignment type, and can be set to either S or P):

$$F_T = \frac{|A_T \cap G_T|}{|A_T|}, \quad R_T = \frac{|A_T \cap G_T|}{|G_T|}, \quad F_T = \frac{2P_T R_T}{P_T + R_T}$$

$$AER = 1 - \frac{|A_P \cap G_S| + |A_P \cap G_P|}{|A_P| + |G_S|}$$

1.4.1 Unlinked words representation

The scores are greatly affected by the representation of NULL links (between a word and no other word: whether they are assigned an explicit link to NULL or removed from the alignments). Explicit NULL links contribute to a higher error rate because in this case the errors are penalised twice: for the incorrect link to NULL and for the missing link to the correct word. Thus both submitted and answer alignments must have the same alignment mode, which can be one of the following:

- **null-align**, where each word is enforced to belong to at least one alignment; if a word doesn’t belong to any alignment, a NULL Possible link is assigned by default.

- **no-null-align**, where all NULL links are removed from both submission and gold standard alignments.
1.4.2 Link weights

In the evaluations of (Och and Ney, 2000; Mihalcea and Pedersen, 2003), each link contributes with the same weight to the count of the various sets. This tends to give more importance to the words aligned in groups than to the words linked only once. To correct this effect, (Melamed, 1998b) proposes to attach a weight to each link. The weight \( w(x, y) \) of a link between two words \( x \) and \( y \) would be inversely proportional to the number of links (\( \text{num}\_\text{links} \)) in which \( x \) and \( y \) are involved:

\[
w(x, y) = \frac{1}{2} \left[ \frac{1}{\text{num}\_\text{links}(x)} + \frac{1}{\text{num}\_\text{links}(y)} \right]
\]

(1)

1.4.3 Implementation

The evaluate function can force the alignments to be both in “null-align” or “no-null-align” mode. The seven measures of the result set (precision, recall and F-measure for sure alignments and for possible alignments, AER rate) are saved in a class called AlignmentEval. You can call it to display a single measure set or a table comparing various result sets.

The evaluate function calculates weighted links if its ‘weighted’ argument is true. The weights must be calculated with respect to the union of the submitted and reference sets. For the measures involving only Sure alignments (\( P_S \) and \( R_S \)), they are calculated with respect to the union of the Sure sets: \( A_S \cup \hat{G}_S \). For the measures mixing Sure and Possible alignments (AER, \( P_P \) and \( R_P \)), the weights are calculated based on the union of the Possible sets: \( A_P \cup \hat{G}_P = A \cup \hat{G} \).

1.5 Symmetrisation

This section concerns Alignment Sets containing asymmetric source-to-target and target-to-source alignments. Combining the source-target and target-source information of the alignments, we can obtain a high precision with low recall alignment (taking the intersection), a low precision with high recall alignment (taking the union), or intermediate combinations. Such intermediate symmetrisation algorithm have been proposed by (Och and Ney, 2003; Koehn et al., 2003; Lambert and Castell, 2004a).

So far the symmetrize function only implements the algorithm described in (Lambert and Castell, 2004a). The algorithm combines two single-word based alignments to produce a symmetric, phrase-based alignment. It exploits the asymmetries in the superposition of the two word alignments to detect the phrases that must be aligned as a whole. The central idea is that if the asymmetry is caused by a language feature such as an idiomatic expression, it will be repeated various times in the corpus, otherwise it will occur only once. So the training must be done in two stages: first, the building of the asymmetries memory. Second, the alignment correction using this memory.
2 Reference

2.1 Alignment Set definition

- new function

<table>
<thead>
<tr>
<th>INPUT parameters</th>
<th>Definition</th>
<th>Rank or Type</th>
<th>optional (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>package</td>
<td></td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>file sets</td>
<td>Ref to an array containing references to each file set (cf below the attributes of a file set). In this version there can only be one file set.</td>
<td>reference string</td>
<td>no</td>
</tr>
</tbody>
</table>

| OUTPUT | Reference to an Alignment Set object |

The input of the new function is a reference to an array of the type (refToFileSet1,refToFileSet2,...). In the present version it is a reference to the array (refToUniqueFileSet). A file set is represented by an array of the type (location,format,range), where:

<table>
<thead>
<tr>
<th>Values</th>
<th>Definition</th>
<th>Rank or Type</th>
<th>optional (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>location</td>
<td>Ref to a hash containing the path and name of the files (or directory for BLINKER). If 'sourceToTarget' is the only entry of the hash, the (string) path can be passed instead of the hash ref.</td>
<td>reference string (see below), or string</td>
<td>no</td>
</tr>
<tr>
<td>format</td>
<td>Format in which the Alignment Set is stored</td>
<td>{'BLINKER','GIZA','NAACL'}</td>
<td>yes ('NAACL')</td>
</tr>
<tr>
<td>range</td>
<td>First and last sentence pairs to be included in Alignment Set</td>
<td>'a-b' where a and b are a number or the empty string</td>
<td>yes ('1-')</td>
</tr>
</tbody>
</table>

The parameters required by each format for the location hash are:

**GIZA:** location=(
  'sourceToTarget' ⇒ source-to-target file path,
  'targetToSource' ⇒ target-to-source file path (optional) )

**NAACL:** location=(
  'source' ⇒ source sentences file path (optional for some functions),
  'target' ⇒ target sentences file path (optional for some functions),
  'sourceToTarget' ⇒ source-to-target alignment file path,
  'targetToSource' ⇒ target-to-source alignment file path (optional) )

**BLINKER:** location=(
  'source' ⇒ source sentences file path (optional for some functions),
  'target' ⇒ target sentences file path (optional for some functions),
  'sourceToTarget' ⇒ directory of source-to-target samp<smb>.SentPair<pn> files,
  'targetToSource' ⇒ directory of target-to-source samp<smb>.SentPair<pn> files (optional) )

'source' and 'target' file names don't need to comply with the Blinker syntax. However, if the source sentence file does comply with it, the sample number is deduced from the name (otherwise it is assumed to be 1). 'sourceToTarget' and 'targetToSource' directory names don't need to respect the blinker notation nor to be situated in the same directory as the sentence files. However the samp<smb>.SentPair<pn> syntax is compulsory.
Note: if "sourceToTarget" is the only entry of the hash, the (string) path can be passed instead of the hash reference.

Code samples:

```
# alternative 1
$location1 = { "source"=>$ENV{ALDIR}."/spanish.naacl",  
"target"=>$ENV{ALDIR}."/english.naacl",  
"sourceToTarget"=>$ENV{ALDIR}."/spanish-english.naacl"};
$fileSet1 = [$location1,"NAACL","1-10"];  
$fileSets = [$fileSet1];
$alSet = Lingua::AlignmentSet->new($fileSets);

# alternative 2
$alSet = Lingua::AlignmentSet->new([[$location1,"NAACL","1-10"];

# alternative 3
$alSet = Lingua::AlignmentSet->new([[ENV{ALDIR}."/spanish-english.naacl",  
"NAACL","1-10"]]);
$alSet->setWordFiles(ENV{ALDIR}."/spanish.naacl",ENV{ALDIR}."/english.naacl"));
```

- **copy method**: creates a new AlignmentSet containing the same data than an existing one, without copying the addresses.

```
my $newAlSet = $alSet->copy;
```

- **setWordFiles method**: sets the sentence files. The two arguments are (sourceFileName, targetFileName). If you first created an AlignmentSet with only the alignment files, use this function before calling subroutines that require the sentence files, like the visualise sub.

- **setSourceFile, setTargetFile, setTargetToSourceFile methods**: same thing for the source, target word files and the targetToSource alignment file, respectively. They take as argument the corresponding file name.

2.2 Conversion between formats

- **chFormat method**:

  Implemented so far: conversion from any format to NAACL or BLINKER format.

  Converts an Alignment Set to the specified format: creates, at the specified location, the necessary file(s) and directory and stores there the Alignment Set in the specified format. It cannot delete the old format files. It starts counting the new Alignment Set sentence pairs from 1, even if in the original Alignment Set they start from a different number.

  If the sentence files (‘source’ and ‘target’ entries of location hash) are present in the old Alignment Set but omitted in the new one, these entries are copied anyway to the new location hash, except if the format is different or if the original Alignment Set range doesn’t start from the first sentence pair. In this last case, a gap could be indeed introduced between the numeration of the sentence and alignment files.
2.3 Visualisation

*visualise method*

This function requires the sentence files (`source` and `target` keys of location hash). You can also use the `setWordFiles` function (see section 2.1) to give the path for these files.

<table>
<thead>
<tr>
<th>INPUT parameters</th>
<th>Definition</th>
<th>Rank or Type</th>
<th>optional (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>alSet</td>
<td>Reference to the input Alignment Set</td>
<td>reference string</td>
<td>no</td>
</tr>
<tr>
<td>representation</td>
<td>Type of visual representation required</td>
<td>{'enumLinks', 'matrix', 'drawLines'}</td>
<td>no</td>
</tr>
<tr>
<td>format</td>
<td>Format of the output file</td>
<td>{'text', 'latex'}</td>
<td>no</td>
</tr>
<tr>
<td>filehandle</td>
<td>Reference to the filehandle of the file where the output should be written</td>
<td>reference string</td>
<td>no</td>
</tr>
<tr>
<td>mark</td>
<td>How a link is marked in the matrix. <code>your_mark</code> is the mark of your choice (but compatible with latex)</td>
<td>{'cross', 'ambiguity', 'confidence', 'your_mark'}</td>
<td>yes (cross)</td>
</tr>
<tr>
<td>alignMode</td>
<td>Take alignment “as is” or force NULL alignment or NO-NULL alignment (see section 1.4)</td>
<td>{'as-is', 'null-align', 'no-null-align'}</td>
<td>yes ('as-is')</td>
</tr>
<tr>
<td>maxRows</td>
<td>Maximum number of rows (source words) allowed in a matrix. If there are more, the alignment is displayed as <code>&quot;enumLinks&quot;, &quot;latex&quot;</code></td>
<td>integers</td>
<td>yes (53)</td>
</tr>
<tr>
<td>maxCols</td>
<td>Maximum number of columns (target words) allowed in a matrix. If there are more, the matrix is continued below</td>
<td>integers</td>
<td>yes (35)</td>
</tr>
</tbody>
</table>
# Example. Creating a LaTeX file with alignment matrices:
open(MAT,'<'.$(ENV[ALDIR]).'/alignments/test.eng-french.tex');
$alSet->visualise("matrix","latex","MAT");
# or, with a personalised mark in matrix:
$alSet->visualise("matrix","latex","MAT","$\blacksquare$");

## Evaluation

- **Evaluate** method: This function integrates the code of Rada Millhaeck's `wa_eval_align.pl` routine (http://www.cs.wt.edu/rada/wpt/code/). If the reference Alignment Set (Gold Standard) and the submission Alignment Set are both in NAACL format and both satisfy the same alignment strategy for NULL alignments (null-align or no-null-align), you can use your files as-is. It will be more efficient. Otherwise you can choose null-align or no-null-align alignment mode to make sure both Alignment Sets are treated in the same way.

The sentence files (source and target entries of location hash) are not taken into account and can be omitted.

<table>
<thead>
<tr>
<th>INPUT parameters</th>
<th>Definition</th>
<th>Rank or Type</th>
<th>optional (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>submissionAllSet</td>
<td>Ref to the Alignment Set to be evaluated</td>
<td>reference string</td>
<td>no</td>
</tr>
<tr>
<td>answerAllSet</td>
<td>Ref to the reference Alignment Set (Gold Standard)</td>
<td>reference string</td>
<td>no</td>
</tr>
<tr>
<td>alignMode</td>
<td>Take alignment as is or force NULL alignment or NO-NULL alignment (see section 1.4)</td>
<td>{&quot;as-is&quot;,&quot;null-align&quot;,&quot;no-null-align&quot;}</td>
<td>yes (&quot;as-is&quot;)</td>
</tr>
<tr>
<td>weighted</td>
<td>If true weights the links according to the method of (Melamed, 1998b)</td>
<td>(0,1)</td>
<td>yes (false)</td>
</tr>
</tbody>
</table>

## Output

- **AlignmentEval class**: This is a hash with the name of the seven measures ([sure][possible] [Precision][Recall][FMeasure] and AER) and their value. You use its methods to display and compare evaluation results:
  - **display** sub: Prints the measures in a table. The arguments are:
    * evaluationMeasure: ref to the evaluationResult object
    * fileHandle: where you want print it (optional, default: STDOUT)
    * format: for now only “text” available (optional)
  - **compare** sub: Prints in a comparative table the results of various evaluations. The arguments are:
    * results: ref to an array of arrays. The first array has one entry for each result set, the second one has 2 entries (the evaluationResult object reference and a string describing the experiment), ie
      [[$refToAlignmentEval1,'description1'], [$refToAlignmentEval2,'description2'], ...]
    * title: A title which appears above the table
    * fileHandle: same as in display
    * format: “text” or “latex” (optional, default “text”)

Code samples for evaluation routines:

```
# Creating an evaluationResult object and pushing it into the evaluation array:
$evaluationResult=$s2e->evaluate($goldStandard,"no-null-align");
push @evaluation,[$evaluationResult,"Non weighted"];```
# Doing the same in one step:
push @evaluation,[\$s2e->evaluate(\$goldStandard,"no-null-align",1),"Weighted"];

# Displaying in a table the two result lines:
Lingua::AlignmentEval::compare(\$evaluation,"My experiments","STDOUT","latex");

2.5 Alignment processing

- **processAlignment** method

  Allows to process the AlignmentSet applying a function to the alignment of each sentence pair of the set. The `Alignment.pm` module contains such functions:

  General subroutines:

  - `forceGroupConsistency`: prohibits situations of the type: if linked(e,f) and linked(e',f) and linked(e',f') but not linked(e,f'). In this case the function links e and f'.
  - `swapSourceTarget`: swaps source and target in the alignments (transforms a link (6 3) in (3 6)).
  - `eliminateWord(wordRegExp,sides)` eliminates a word (defined as a regular expression) from a side of the corpus (“source” or “target”) and updates the links accordingly. This is a contribution from A. de Gisbert and J. M. Grego.

Subroutines which combine source-target and target-source alignments:

- `intersect`
- `getUnion`
- `selectSideWithLeastLinks`: for each sentence pair, selects, from source-target and target-source alignment, that with the highest number of individual links.
- `selectSideWithMostLinks`

<table>
<thead>
<tr>
<th>INPUT parameters</th>
<th>Definition</th>
<th>Rank or Type</th>
<th>optional (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>allSet</td>
<td>Reference to the input Alignment Set</td>
<td>reference string</td>
<td>no</td>
</tr>
<tr>
<td>AlignmentSub</td>
<td>Name of a subroutine of the Alignment.pm module. This subroutine is applied to each sentence pair. If the subroutine takes N arguments, AlignmentSub is a reference to the following array: (sub name, arg1, ... , argN)</td>
<td>string or reference string</td>
<td>no</td>
</tr>
<tr>
<td>location</td>
<td>Ref to a hash containing the path and name of the files in the NEW format. If “sourceToTarget” is the only entry of the hash, the (string) path can be passed instead of the hash ref</td>
<td>reference string (cf section 2.1), or string</td>
<td>no</td>
</tr>
<tr>
<td>format</td>
<td>Required new format</td>
<td>{'BLINKER','GIZA','NAACL'}</td>
<td>yes ('NAACL')</td>
</tr>
<tr>
<td>alignMode</td>
<td>Take alignment &quot;as is&quot; or force NULL alignment or NO-NULL alignment (see section 1.4)</td>
<td>{'as-is', 'null-align', 'no-null-align'}</td>
<td>yes ('as-is')</td>
</tr>
</tbody>
</table>

**OUTPUT** Reference to an Alignment Set object

As mentioned for the chFormat sub, if the original allSet object contained the sentences files, the corresponding entries of the location hash are conserved, except if the format is
# create the union of source-target and target-source alignments
$s2e = Lingua::AlignmentSet->new({"sourceToTarget" => $spa2engTest,
                           "targetToSource" => $eng2spaTest,"GIZA"});
my $union=$s2e->processAlignment("lingua::Alignment::getUnion","union.naacl","NAACL");

# remove ?!. signs from the Blinker reference corpus and save it as Naacl file:

#1 remove from target side of the corpus:
my $answer = Lingua::AlignmentSet->new({"data/answer/spanish-english","BLINKER"]);
# need to add the target words file because we remove from that side:
$answer->setTargetFile("data/answer/english.blinker");
# define output location:
$newLocation = {"target"=>"data/english-without.naacl",
               "sourceToTarget"=>"data/spanish-english-interm.naacl");
my $refToArray = ["Lingua::Alignment::eliminateWord",'.\?\!\!',"target"];
my $output = $answer->processAlignment($refToArray,$newLocation);

#2 Remove row from source side:
# we take as input alignment the previous one (already removed in target side).
# however, to remove from source, we need to add the source words:
$output->setSourceFile("data/spanish.naacl");
# define output location:
$newLocation = {"source"=>"data/spanish-without.naacl",
               "sourceToTarget"=>"data/spanish-english-without.naacl");
$refToArray = ["Lingua::Alignment::eliminateWord",'.\?\!\!',"source"];
$output = $output->processAlignment($refToArray,$newLocation);

**symmetrize method**

<table>
<thead>
<tr>
<th>INPUT parameters</th>
<th>Definition</th>
<th>Rank or Type</th>
<th>optional (default)</th>
</tr>
</thead>
<tbody>
<tr>
<td>alSet</td>
<td>Reference to the input Alignment Set</td>
<td>reference string</td>
<td>no</td>
</tr>
<tr>
<td>location</td>
<td>Ref to a hash containing the path and name of the files in the NEW format. If &quot;sourceToTarget&quot; is the only entry of the hash, the (string) path can be passed instead of the hash ref</td>
<td>reference string (cf section 2.1), or string</td>
<td>no</td>
</tr>
<tr>
<td>format</td>
<td>Required new format</td>
<td>{'BLINKER','GIZA','NAACL'}</td>
<td>yes ('NAACL')</td>
</tr>
<tr>
<td>groupsDir</td>
<td>Directory of the asymmetric phrases memory files 'groups' and 'sub-Groups'</td>
<td>directory string</td>
<td>yes (&quot;&quot;)</td>
</tr>
<tr>
<td>selectPhrases</td>
<td>If true, it selects the asymmetric phrases and save them in the &quot;$groupsDir/groups&quot; (and 'sub-Groups') files and updates the alignment. If false, it only updates the alignment (using the existing memory).</td>
<td>boolean</td>
<td>yes (false)</td>
</tr>
<tr>
<td>options</td>
<td>Reference to a hash of options</td>
<td>reference string</td>
<td>yes</td>
</tr>
</tbody>
</table>
The defaults of the options hash are pretty reasonable. However you might want to give a higher value to "minPhraseFrequency" if you have many data. You can get more recall or precision changing the "defaultActionGrouping" option. The options hash has the following entries (for more details see (Lambert and Castell, 2004b)):

- **minPhraseFrequency** (default 2) The minimum number of occurrences necessary to select a phrase in the memory.
- **onlyGroups** (default 1) In the alignment update stage, considers only phrases of the 'groups' file (that are strictly asymmetric zones of the alignment combination) or considers also phrases of the 'subGroups' file (which are subgroups of the phrases in the 'groups' files).
- **defaultActionGrouping** (default "Lingua::Alignment::getUnion") Action to take if there is no applicable phrase in the memory.
- **defaultActionGeneral** (default "Lingua::Alignment::intersect") Action to take if the asymmetric zone is too small or too big to be reasonably linked as a group (normally the best is to take the intersection to avoid a drop in precision).

2.6 Miscellaneous

- **chooseSubsets** method Returns a randomly chosen list (in random order) of line Numbers contained in the AlignmentSet object. To sort this list, do:

  ```perl
  my @sortedSelection = sort { $a <=> $b; } @selection;
  ```

  chooseSubsets takes an additional argument, the size of the desired subset.

- **getSize** method Method which calculates the number of sentence pairs in the Alignment Set.

3 Known problems

By definition of the Alignment Set, the numeration in the input and output files can be distinct. Indeed, the sentence pairs of an Alignment Set could in theory be stored in various file sets.

If the range of your Alignment Set doesn't start from the first sentence pair and your are converting or processing the alignments, keep in mind that the numeration after conversion will be different from that before the conversion.

4 To Do List

If you need a tool which is not present yet in this library, why not consideringincluding it? Examples of further developments of the library could be:

- Allow an Alignment Set to be contained in various file sets.
- Implement the drawLines way of visualising an alignment (see section 1.3), which is the most intuitive.
- Implement other symmetrisation methods.
- Add other types of evaluation measures, which wouldn't have the limitations of the AER (see (Lambert and Castell, 2004a)).
- Instead of reading the files sentence per sentence, read them memory chunk per memory chunk
5 Acknowledgements

The author want to thank Adrià de Gispert for testing the initial versions of the library and for his helpful comments. This library includes some code from Rada Mihalcea, Josep Maria Crego and Adrià de Gispert. This work has been (partially) granted by the Spanish Government under grant TIC2002-04447-C02.

References


Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Research Reports - 2004

- LSI-04-2-R: *Comparison of Methods to Predict Ozone Concentration*, Orozco, J.
- LSI-04-3-R: *Towards the definition of a taxonomy for the cots product’s market*, Ayala, Claudia P.
- LSI-04-4-R: *Modelling Coalition Formation over Time for Iterative Coalition Games*, Mérida-Campos, C. and Willmott, S.
- LSI-04-6-R: *An Analysis Pattern for Electronic Marketplaces*, Queralt, A. and Teniente, E.
- LSI-04-7-R: *Exploring Dopamine-Mediated Reward Processing through the Analysis of EEG-Measured Gamma-Band Brain Oscillations*, Vellido, A. and El-Deredy, W.
- LSI-04-15-R: *Evaluation and symmetrisation of alignments obtained with the Giza++ software*, Lambert, P. and Castell, N.
- LSI-04-16-R: *A note on the use of topology extensions for provoking instability in communication networks*, Blesa, M.J.
- LSI-04-19-R: *Adding Efficient and Reliable Access Paths to the JCF*, Marco, J. and Franch, X.
• LSI-04-20-R: *Exploiting Simple Corporate Memory in Iterative Coalition Games*, Mérida-Campos, C. and Willmott, S.
• LSI-04-22-R: *Complexity issues on bounded restrictive H-coloring*, Diaz, J. and Serna, M. and Thilikos, D.M.
• LSI-04-23-R: *Chromatic number in random scaled sector graphs*, Díaz, J. and Sanwalani, V. and Serna, M. and Spirakis, P.
• LSI-04-24-R: *Bounds on the bisection width for random d-regular graphs*, Díaz, J. and Serna, M. and Wormald, N.C.
• LSI-04-25-R: *Open Source environment to define constraints in route planning for GIS-T*, Pérez, L. and Silveira, A. da M.
• LSI-04-26-R: *A basic repository of operations for the refinement of general ontologies*, de Palol, X.
• LSI-04-27-R: *Tetrahedral mesh subdivision based on underlying volume data*, Rodríguez, L. and Navazo, I. and Vinacua, A.
• LSI-04-28-R: *The Price of Connectedness in Expansions*, Fomin, F.V. and Fraigniaud, P. and Thilikos, D.M.
• LSI-04-29-R: *Smaller kernels for hitting set problems of constant arity*, Nishimura, N. and Ragde, P. and Thilikos, D.M.
• LSI-04-30-R: *Searching Spatial Sense in the Ontological World: Discovering Spatial Objects*, Morocho, V and Pérez, L. and Saltor, F.
• LSI-04-33-R: *Multisided patches*, Pla, N. and Vigo, M. and Cotrina, J.
• LSI-04-34-R: *SVMTool: A general POS tagger generator based on Support Vector Machines*, Giménez, J. and Márquez, LL.
• LSI-04-35-R: *A distributed and mobile component system based on the ambient calculus*, Mylonakis, N. and Orejas, F.
• LSI-04-36-R: *Developing Competitive HMM PoS Taggers Using Small Training Corpora*, Padró, M. and Padró, LL.
• LSI-04-37-R: *The AlignmentSet Toolkit*, Lambert, P.

Hardcopies of reports can be ordered from:

Núria Sanchez
Departament de Llenguatges i Sistemes Informàtics
See also the Departament WWW pages, http://wwwlsi.upc.es/