Documentation for the Alignment Set Toolkit

Patrik Lambert (lambert@gps.tsc.upc.es)

May 17, 2006

Version 1.1

Contents

1 Description 2
- 1.1 The Alignment Set . . . 2
- 1.2 Description of storage formats . . . 2
- 1.3 Visualisation representations and formats . . . 5
- 1.4 Evaluation . . . 6
  - 1.4.1 Unlinked words representation . . . 7
  - 1.4.2 Link weights . . . 7
  - 1.4.3 Implementation . . . 7
- 1.5 Symmetrisation . . . 7

2 Reference 8
- 2.1 Command-line Tools . . . 8
- 2.2 Alignment Set definition . . . 8
  - `new` function . . . 8
- 2.3 Conversion between formats . . . 9
  - `chFormat` method . . . 9
- 2.4 Visualisation . . . 10
  - `visualise` method . . . 10
- 2.5 Evaluation . . . 11
  - `evaluate` method . . . 11
  - The `AlignmentEval` class . . . 12
- 2.6 Alignment processing . . . 12
  - `processAlignment` method . . . 12
  - `orderAsBilCorpus` method . . . 14
  - `adaptToBilCorpus` method . . . 14
  - `symmetrize` method . . . 14
- 2.7 Miscellaneous . . . 15
  - `chooseSubsets` method . . . 15

3 Known problems 15

4 To Do List 16

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 section: 2.2):

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 2.2.
- **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.

TALP This is the default format in version 1.1 (but not in version 1.0). The Alignment Set is stored in three separate files: one for the source sentences, one for the target sentences and one for the alignments. In each file, each line corresponds to one sentence pair and the sentence pair order is the same in all files. In this way source sentence, target sentence and links of the same sentence pair share the same line number. Source and target sentences are stored as plain text, and links as a sequence of position numbers (starting from 1, as 0 is reserved for NULL word) separated by "s" or a dash (sure links), or a p (possible links). Example:

source words file

```
I can not say anything at this stage .
We will consider the matter .
```

target words file

```
Así , de momento , no puedo pronunciarme .
Deberemos examinar la cuestión .
```

source-target alignment file

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

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 día dieciocho , francamente es del todo imposible , no le 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 }) francamente ({ 6 7 8 }) es ({ }) del ({ }) todo ({ }) imposible ({
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):

```
0008 4 2 S
0008 1 1 P
0008 2 1 P
0008 3 1 P
0009 1 1 S
0009 2 3 S
0009 3 4 S
0009 4 5 S
0009 5 6 S
0009 8 9 S
0009 9 10 S
0009 10 11 S
0009 11 13 S
0009 12 15 S
0009 13 16 S
0009 2 2 P
0009 6 7 P
0009 6 8 P
0009 7 7 P
0009 7 8 P
0009 11 14 P
0009 12 14 P
0009 0 12 P
```

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<anb>` (`<anb>` 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 2.2).

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<pnb>` **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: every 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).

4
NULL ¿ cuántas personas van ?
NULL how many people are travelling ?
¿ → how
cuántas ← how
cuántas ↔ many
personas ↔ people
van ← are
van ↔ travelling
? ↔ ?

Figure 1: `enumLinks` representation, `latex` format

matrix Writes 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.

4
NULL ¿ cuántas personas van ?
NULL how many people are travelling ?

?	.	.	.	.	.	.	+
van	.	.	.	.	−	+	.
personas	.	.	.	+	.	.	.
cuántas	.	−	+	.	.	.	.
¿	.	−	.	.	.	.	.
NULL	.	.	.	.	.	.	.
	NULL	how	many	people	are	travelling	?

Figure 2: `matrix` representation, `latex` format, `cross` mark

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.

Figure 3: `drawLines` representation

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 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 $\mathcal{A}_\mathcal{S}$, $\mathcal{A}_\mathcal{P}$ and $\mathcal{G}_\mathcal{S}$, $\mathcal{G}_\mathcal{P}$, 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 $\mathcal{A_S} \subseteq \mathcal{A_P}$ and $\mathcal{G_S} \subseteq \mathcal{G_P}$. The following measures are defined (where T is the alignment type, and can be set to either S or P):

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

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

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 $w(\mathrm{x},\mathrm{y})$ of a link between two words x and y would be inversely proportional to the number of links (*num_links*) in which x and y are involved:

$$w(\mathrm{x},\mathrm{y}) = \frac{1}{2}\left[\frac{1}{num\_links(\mathrm{x})} + \frac{1}{num\_links(\mathrm{y})}\right] \quad (1)$$

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: $\mathcal{A_S} \cup \mathcal{G_S}$. 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: $\mathcal{A_P} \cup \mathcal{G_P} = \mathcal{A} \cup \mathcal{G}$.

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 Command-line Tools

Important: a series of command-line tools are available, with internal documentation (which is displayed adding the option -man). These tools encapsulate calls to the perl library and do the same as the subroutines described here, without the need to read the following reference and to write perl code.

2.2 Alignment Set definition

- `new` function

INPUT parameters	**Definition**	**Rank or Type**	**optional (default)**
`package`			no
`file sets`	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.	reference string	no
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:

Values	**Definition**	**Rank or Type**	**optional (default)**
`location`	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	reference string (see below), or string	no
`format`	Format in which the Alignment Set is stored	{'BLINKER','GIZA', 'NAACL','TALP'}	yes ('TALP')
`range`	First and last sentence pairs to be included in Alignment Set	'a-b' where a and b are a number or the empty string	yes ('1-')

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 and TALP: `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<snb>.SentPair<pnb>` files,
`‘‘targetToSource’’`	⇒ directory of target-to-source `samp<snb>.SentPair<pnb>` 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<snb>.SentPair<pnb>` 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.

2.3 Conversion between formats

- `chFormat` method :

  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.

INPUT parameters	**Definition**	**Rank or Type**	**optional (default)**
`alSet`	Reference to the input Alignment Set	reference string	no
`location`	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	reference string (cf section 2.2), or string	no
`format`	Required new format	{'BLINKER','GIZA', 'NAACL','TALP'}	yes ('TALP')
`alignMode`	Take alignment "as is" or force NULL alignment or NO-NULL alignment (see section 1.4)	{'as-is', 'null-align', 'no-null-align'}	yes ('as-is')

```
# convert from GIZA to NAACL format
my $spa2eng = Lingua::AlignmentSet->new([["$ENV{ALDIR}/al.giza.eng2spa","GIZA","1-15"]]);

my $newLocation = {"source"=>$ENV{ALDIR}."/naacl-source.1.1-15",
                   "target"=>$ENV{ALDIR}."/naacl-target.1.1-15",
                   "sourceToTarget"=>$ENV{ALDIR}."/naacl-al-st.1.1-15"};
$spa2eng->chFormat($newLocation,"NAACL");
```

2.4 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 2.2) to give the path for these files.

INPUT parameters	**Definition**	**Rank or Type**	**optional (default)**
`alSet`	Reference to the input Alignment Set	reference string	no
`representation`	Type of visual representation required	{’enumLinks’, ’matrix’, ’drawLines’}	no
`format`	Format of the output file	{’text’, ’latex’}	no
`filehandle`	Reference to the filehandle of the file where the output should be written	reference string	no
`mark`	How a link is marked in the matrix. *your_mark* is the mark of your choice (but compatible with latex)	{’cross’, ’ambiguity’, ’confidence’, *your_mark*}	yes (cross)
`alignMode`	Take alignment “as is” or force NULL alignment or NO-NULL alignment (see section 1.4)	{’as-is’, ’null-align’, ’no-null-align’}	yes (’as-is’)
`maxRows`	Maximum number of rows (source words) allowed in a matrix. If there are more, the alignment is displayed as `‘‘enumLinks’’,’’latex’’`	integers	yes (53)
`maxCols`	Maximum number of columns (target words) allowed in a matrix. If there are more, the matrix is continued below	integers	yes (35)

```
# 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$’);
```

2.5 Evaluation

- `evaluate` method This function integrates the code of Rada Milhalcea `wa_eval_align.pl` routine (http://www.cs.unt.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.

INPUT parameters	**Definition**	**Rank or Type**	**optional (default)**
`submissionAlSet`	Ref to the Alignment Set to be evaluated	reference string	no
`answerAlSet`	Ref to the reference Alignment Set (Gold Standard)	reference string	no
`alignMode`	Take alignment “as is” or force NULL alignment or NO-NULL alignment (see section 1.4)	{’as-is’,’null-align’,’no-null-align’}	yes (’as-is’)
`weighted`	If true weights the links according to the method of (Melamed, 1998b)	{0,1}	yes (false)
OUTPUT	Reference to an evaluation result object		

- `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");
```

2.6 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)).
  - `regexpReplace`: Substitutes, in a side of the corpus, a string (defined by a regular expression) by another and updates the links accordingly. There are 3 arguments: the regular expressions (pattern and replacement) and the side (source or target) (see the man examples). Notes:
    - In case of deleting various words, all added words are linked to all positions to which deleted words were linked. $\$al->sourceLinks$ information can be lost for replaced words.
    - The regexp is applied to the side of the corpus, and the smallest set of additions and deletions necessary to turn the original word sequence into the modified one is computed using algorithm::diff. In practice, this set is not always minimal, and in these cases various words are replaced by various so links may be changed. To avoid this problem use `replaceWords` subroutine.
    - Is more eficient in ”source” side than in ”target” side.

- `replaceWords`: Substitutes, in a side of the corpus, a string (of words separated by a white space) by another and updates the links accordingly. There are 3 arguments: the string of words to be replaced, the string of replacement words and the side (source or target) (see the man examples). Notes:
  - In case of deleting various words, all added words are linked to all positions to which deleted words were linked. $\$al- > sourceLinks$ information can be lost for replaced words.
  - Is more eficient in "source" side than in "target" side.
- `manyToMany2joined`: introduces underscore between links of many-to-many groups in source to target alignment. Warning: this subroutine only changes words files, not the links file.
- `joined2ManyToMany`: recreates links of words linked by underscore and removes underscores
- `getAlClusters`: gets the alignment as clusters of positions aligned together.

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`

INPUT parameters	**Definition**	**Rank or Type**	**optional (default)**
`alSet`	Reference to the input Alignment Set	reference string	no
`AlignmentSub`	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)	string or reference string	no
`location`	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	reference string (cf section 2.2), or string	no
`format`	Required new format	{'BLINKER','GIZA', 'NAACL','TALP'}	yes ('TALP')
`alignMode`	Take alignment "as is" or force NULL alignment or NO-NULL alignment (see section 1.4)	{'as-is', 'null-align', 'no-null-align'}	yes ('as-is')
OUTPUT	Reference to an Alignment Set object		

As mentioned for the `chFormat` sub, if the original alSet object contained the sentences files, the corresponding entries of the location hash are conserved, except if the format is different or if the range doesn't start with the first sentence pair.

```
# 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::replaceWords",'\.|\?|!','',"target"];
  my $output = $answer->processAlignment($refToArray,$newLocation);

#2 Remove now 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::replaceWords",'\.|\?|!','',"source"];
  $output = $output->processAlignment($refToArray,$newLocation);
```

- `orderAsBilCorpus` method Place sentence pairs of a secondary corpus at the head of the Alignment Set, in the same order. See more details with the command-line tool (`perl orderAlSetAsBilCorpus.pl -man`).

- `adaptToBilCorpus` method Looks if the Alignment Set sentence pairs are in another bilingual corpus, and for each sentence pair which is not in the corpus, it searches the corpus sentence pair with best longuest common subsequence (LCS) ratio. Finally, it detects the edits (word insertions, deletions, and substitutions) neces- sary to pass from the Alignment Set sentences to the corpus sentences with best LCS ratio, prints the edit list and transmits these edits in the output links file. See more details with the command-line tool (`perl adaptAlSetToBilCorpus.pl -man`).

- `symmetrize` method

INPUT parameters	**Definition**	**Rank or Type**	**optional (default)**
`alSet`	Reference to the input Alignment Set	reference string	no
`location`	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	reference string (cf section 2.2), or string	no
`format`	Required new format	{’BLINKER’,’GIZA’, ’NAACL’,’TALP’}	yes (’TALP’)
`groupsDir`	Directory of the asymmetric phrases memory files ’groups’ and ’subGroups’	directory string	yes (””)
`selectPhrases`	if true, it selects the asymmetric phrases and save them in the ”$groupsDir/groups” (and ’subGroups’) files and updates the alignment. If false, it only updates the alignment (using the existing memory).	boolean	yes (false)
`options`	Reference to a hash of options	reference string	yes

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).

2.7 Miscellaneous

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

  ```
  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 considering including 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)).

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. This work has been (partially) granted by the Spanish Government under grant TIC2002-04447-C02.

References

Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, D. Melamed, F. J. Och, D. Purdy, N. A. Smith, and D. Yarowsky. 1999. Statistical machine translation. Technical report, John Hopkins University Summer Workshop. http://www.clsp.jhu.edu/ws99/projects/mt/.

P. Koehn, F.J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of the 41th Annual Meeting of the Association for Computational Linguistics.*

P. Lambert and N. Castell. 2004a. Alignment of parallel corpora exploiting asymmetrically aligned phrases. In *Proc. of the LREC 2004 Workshop on the Amazing Utility of Parallel and Comparable Corpora*, Lisbon, Portugal, May 25.

Patrik Lambert and Núria Castell. 2004b. Evaluation and symmetrisation of alignments obtained with the giza++ software. Technical Report LSI–04–15–R, Technical University of Catalonia. http://www.lsi.upc.es/dept/techreps/techreps.html.

I. Dan Melamed. 1998a. Annotation style guide for the blinker project. Technical Report 98-06, IRCS.

I. Dan Melamed. 1998b. Manual annotation of translational equivalence. Technical Report 98-07, IRCS.

Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In Rada Mihalcea and Ted Pedersen, editors, *HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–10, Edmonton, Alberta, Canada, May 31. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2000. A comparison of alignment models for statistical machine translation. In *Proc. of the 18th Int. Conf. on Computational Linguistics*, pages 1086–1090, Saarbrucken,Germany, August.

F.J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, March.

Franz Josef Och. 2000. Giza++: Training of statistical translation models. http://www.isi.edu/~och/GIZA++.html.

Ted Pedersen and Brian Rassier. 2003. Aligner for parallel corpora. http://www.d.umn.edu/~tpederse/parallel.html.