Leveraging the Advantages of Associative Alignment Methods for PB-SMT Systems*

Baosong Yang and Yves Lepage**

IPS, Waseda University
yves.lepage@waseda.jp

Abstract. Training statistical machine translation systems used to require heavy computation times. It has been shown that approximations in the probabilistic approach could lead to impressing improvements (Fast align). We show that, by leveraging the advantages of the associative approach, we achieve similar, even faster, training times, while keeping comparable BLEU scores. Our contributions are of two types: of the engineering type, by introducing multi-processing both in sampling-based alignment and hierarchical sub-sentential alignment; of modeling type, by introducting approximations in hierarchical sub-sentential alignment that lead to important reductions in time without affecting the alignments produced. We test and compare our improvements on six typical language pairs of the Europarl corpus.

1 Introduction

Sub-sentential alignment, computed based on word associations, is the core of the training process in Phrase-based Statistical Machine Translation (PB-SMT). These two processes are crucial for the accuracy of translation, but they are also very time-consuming.

The IBM models [2] and the grow-diag-final-and heuristic are the most popular approach. They have been integrated as the GIZA++ tool [16], or MGIZA++ [6] for a parallel implementation, in the PB-SMT toolkit Moses[1]. A log-linear re-parameterisation of IBM Model 2 has been implemented in Fast align[2] [4]. It led to much faster training times.

IBM models are probabilistic models, so that the optimisation process requires the knowledge of the entire parallel corpus to estimate the parameters [15]. On the contrary, *associative* methods, as characterised in [5], do not rely on a global alignment model, but use local maximisation so that each sentence

* This paper is a part of the outcome of research performed under a Waseda University Grant for Special Research Projects (Project number: 2015A-063).

** Thanks to Chonlathorn Kwankajornkiet from Chulalongkorn University, Thailand, for her contribution in implementing the C core component of Cutnalign during a training period at IPS, Waseda University.

[1] http://www.statmt.org

[2] http://github.com/clab/fast_align

pair can be processed independently. Various criteria may be used like Dice coefficient, cosine, mutual information [5] or likelihood ratio [3, 17].

Sampling-based multilingual alignment, introduced in [14], and implemented as Anymalign[3], is an associative method for the computation of word associations. The method repeatedly draws random (mainly small) sub-corpora from the parallel corpus and obtains occurrence distributions of word pairs (or short word sequence pairs) within each sub-corpus so as to ultimately produce a word association table.

Bilingual hierarchical sub-sentential alignment, introduced in [13], and implemented as Cutnalign[4], is an associative method to compute sub-sentential alignments. It processes parallel sentences using a recursive binary segmentation of the alignment matrix. It yields performance comparable with that of state-of-the-art methods [7].

Figure 1 describes the training process which combines these two associative methods: it replaces GIZA++ and the grow-diag-final-and heuristic: Cutnalign uses word associations produced by Anymalign as input, and outputs sub-sentential alignments. The relevant script in Moses[5] then extracts phrases from sub-sentential alignments.

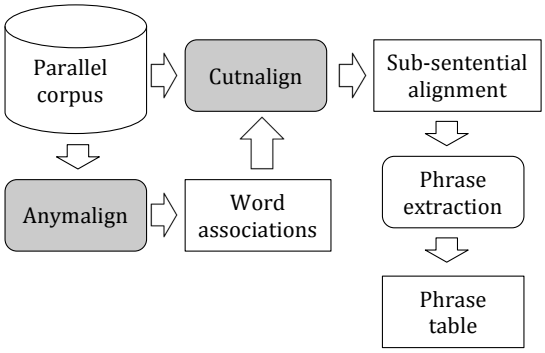

Fig. 1. Combination of two associative methods, Anymalign and Cutnalign, to obtain phrase tables from a parallel corpus

We present various types of improvements in the current implementations of the two above-mentioned associative methods that make them competitive with recent probabilistic approaches.

– improvement of the engineering type: we exploit the essence of associative methods and introduce multi-processing in both sampling-based alignment

[3] `https://anymalign.limsi.fr/`
[4] Thanks to the authors for providing the source code.
[5] `train-model.perl --first step 4`

and hierarchical sub-sentential alignment so as to trivially accelerate the overall alignment process. To compare with baseline systems, which are implemented in C/C++, in a more fair way, we also reimplement the core component of Cutnalign in C;
- improvement of the modelling type: we propose approximations to accelerate some decisions and a method to reduce the search space in hierarchical sub-sentential alignment so that additional speed-ups are obtained.

The rest of this paper is organized as follows: Section 2 rapidly describes the engineering improvements obtained by eliminating unnecessary computations and by introducing multi-processing. Section 3 explains and justifies in practice three practical and empirical improvements in hierarchical sub-sentential alignment. Section 4 gives an incremental evaluation of our work and a comparison with state-of-the-art methods on a series of machine translation experiments.

2 Acceleration thanks to Re-Engineering

2.1 Elimination of Unnecessary Computations

The first implementation of Cutnalign made use of a different matrix than the sentence pair matrix to accelerate computation. In that matrix, each cell contains the sum of all translation strengths of the block starting in the top left corner of the sentence pair matrix up to that cell, i.e., each cell (i, j) contains $W(X_{0...i}, Y_{0...j})$. As the process is recursive, the computation of W for many blocks inside the matrix is required. For a block $(X_{i_1...i_2}, Y_{j_1...j_2})$ extending from i_1 to i_2 and from j_1 to j_2, its corresponding $W(X_{i_1...i_2}, Y_{j_1...j_2})$ is easily computed as $W(X_{0...i_2}, Y_{0...j_2}) - W(X_{0...i_1}, Y_{0...j_1})$, thus saving computation time.

For generalisation purposes and readability of coding, the code introduced the factor $W(X_{0...0}, Y_{0...0})$ to be subtracted when computing W. This was hidden in one elegant general case by the use of general indices, that unfortunately could take the value 0 at some point during computation. As $W(X_{0...0}, Y_{0...0})$ is equal to 0, unnecessary subtractions by 0 were performed.

We isolated and rewrote nine different sub-cases (to the detriment of the aesthetics of the code) so as to eliminate such unnecessary subtractions. Much to our surprise, this led to an acceleration by a factor of approximately 40 times. The line labelled S in Table 2 reports such improvements.

2.2 Multi-Processing

With the ubiquity of multi-processor systems, any software tool should allow optimal use of computer resources whenever possible. Associative methods make it possible by construction.

Word Associations Anymalign draws random sub-corpora from the training corpus, and computes the occurrence distribution profiles for all words over all

sentence pairs in each sub-corpus. Consequently, the process for each sub-corpus is independent. Thanks to this characteristic, no data needs to be transferred for synchronisation, thus avoiding any time consumption overhead usually observed when I/O operations are extensively used. The sizes of the sub-corpora are randomly drawn according to a specific distribution. Consequently, sampling of sizes can also be performed independently in different sub-processes, without affecting the general behavior in any way. Multi-processing is thus done by having each sub-process randomly drawing sub-corpora sizes, drawing sub-corpora of the given sizes, and computing word associations. After the master process has received an interruption[6], word associations and their associated frequencies are output by each sub-process and passed over to the master process which sums up the frequencies of each word association produced by each sub-process and computes association scores.

Experiments show that only very small, and insignificant differences in associations output exist between the mono-processing and multi-processing versions. They are due to differences in sampling.

Table 1 gives the BLEU scores obtained when allotting 15 minutes to Anymalign on one core, and 5 minutes on four cores. No significant difference is observed.

Table 1. No significant difference in BLEU scores is observed when using the mono-processor (original) or the multi-processor (M) versions of Anymalign to compute word associations. The number of word associations differs by 11% = (575,641 - 517,274) / 517,274. The data used are described in Sec. 4.1.

Anymalign	time (min)	# of word assoc.	BLEU (%)
original	15	517,274	34.0 ± 0.8
M	5	575,641	34.1 ± 0.8

Hierarchical Sub-sentential Alignment Cutnalign is easily parallelised by observing that the sub-sentential alignment process for each different sentence pair is independent from the other ones. The first two lines in Table 2 show the times obtained on increasing amounts of sentence pairs using the same word associations output by Anymalign. Using 4 cores divides the time by 3. Experiments have shown that using 4 cores divides the time by 3.

By design, introducing multi-processing as described above does not affect the quality of the final results, because the parallelised and non-parallelised implementations are theoretically equivalent. We checked that sub-sentential alignments outputs in both implementations are exactly the same.

[6] Anymalign is an anytime process, and should be given a timeout.

Table 2. Times (in minutes) for different versions of Cutnalign and different numbers of sentence pairs. The numbers in parentheses give the speed-up. The first line is the original implementation. M means a multi-processor version (see Sect. 2.2, 4 cores here). S means a version avoiding unnecessary subtractions of zeros (see Sect. 2.1). Same data as for Table 1

# of sentence pairs	2,500	5,000	7,500	10,000
original	62 (× 1)	141 (× 1)	212 (× 1)	288 (× 1)
M	19 (× 3)	45 (× 3)	64 (× 3)	95 (× 3)
S	1.5 (× 41)	4 (× 35)	5 (× 42)	7 (× 41)
M+S	0.5 (×124)	1 (×141)	1.5 (×141)	2 (×144)

3 Two Approximations in Hierarchical Sub-sentential Alignment

The original sub-sentential alignment method proposed in [13] can be explained in three main steps.

First, it builds a sentence pair matrix for a given sentence pair where the translation strength between a source word s and a target word t is computed as the product of the two association scores $p(s|t)$ and $p(t|s)$. In their proposal, as well as in this paper, the association scores are computed by Anymalign. Figure 2 illustrates such a sentence pair matrix. Notice that the content of the cells in the sentence pair matrix is bidirectional by construction.

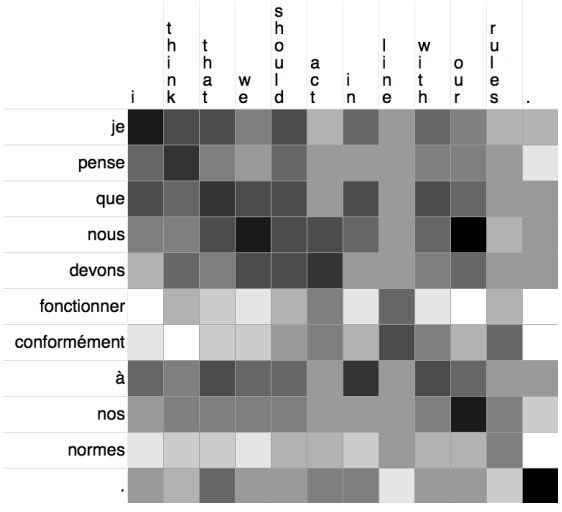

Fig. 2. Translation strengths in a French–English sentence pair matrix. Cells are grayed from 0.0 (white) to 1.0 (black) on a logarithmic scale.

Then, the method searches for the best alignment by computing the best segmentation of the sentences into sub-blocks recursively. This is done by computing the optimal bi-clustering of a bipartite graph, as suggested in the work of [18] for document clustering. For this purpose, a score named cut (see Equation 1) is computed that sums up all the cells in the two sub-blocks of a block in the sentence pair matrix (see Fig. 3). In the definition of cut, $W(X,Y)$ is the sum of all translation strengths between all source and target words inside a sub-block (X,Y).

$$\text{cut}(X,Y) = W(X,\overline{Y}) + W(\overline{X},Y) \tag{1}$$

In order to make the partition as dense as possible, [18] use a normalized variant of the score named Ncut (see Equation 2)[7]. The best segmentation minimizes this variant over all $\text{Ncut}(X,Y)$ and $\text{Ncut}(\bar{X},Y)$, thus making simultaneously the decision of where to split and in which direction.

$$\begin{aligned} \text{Ncut}(X,Y) = & \quad \frac{\text{cut}(X,Y)}{\text{cut}(X,Y) + 2\times W(X,Y)} \\ + & \quad \frac{cut(\bar{X},\bar{Y})}{\text{cut}(\bar{X},\bar{Y}) + 2\times W(\bar{X},\bar{Y})} \end{aligned} \tag{2}$$

Finally, as the method recursively segments the matrix, an alignment between the pair of sentences is obtained when no block remains to segment.

3.1 Decision on the Direction First

When splitting a block inside the sentence pair matrix into two sub-blocks, the segmentation method makes two theoretically separate decisions:

- which location, i.e., where to split, e.g., after (devons, act) in Fig. 3, and
- which direction, linear or cross, i.e., choosing either the black segmentation or the white one in Fig. 3.

The original approach consists in making the two decisions simultaneously, by selecting the max over all possible $\text{Ncut}(X,Y)$ and $\text{Ncut}(\bar{X},Y)$. For a block of size $N \times M$, there are $2 \times N \times M$ Ncuts to compute. The original implementation of Cutnalign adopts this approach.

Our approach will separate the two decisions. We will first decide the direction and then the location. In practice, the use of cut instead of Ncut allows to make the decision on the direction without much difference in the final segmentation result. This leads to a reduction in computation because the computation of Ncut requires the computation of cut: making the decision in advance on cut avoids the computation of Ncut for the other direction. In this way, only half of the Ncuts, i.e., $N \times M$, are computed. As only one location inside a block is selected afterwards, possibly incorrect decisions on directions do remain unseen, and the final segmentation is not affected by them.

[7] Notice that, by definition: $\text{Ncut}(X,Y) = \text{Ncut}(\bar{X},\bar{Y})$ and $\text{Ncut}(X,\bar{Y}) = \text{Ncut}(\bar{X},Y)$. The same holds for cut.

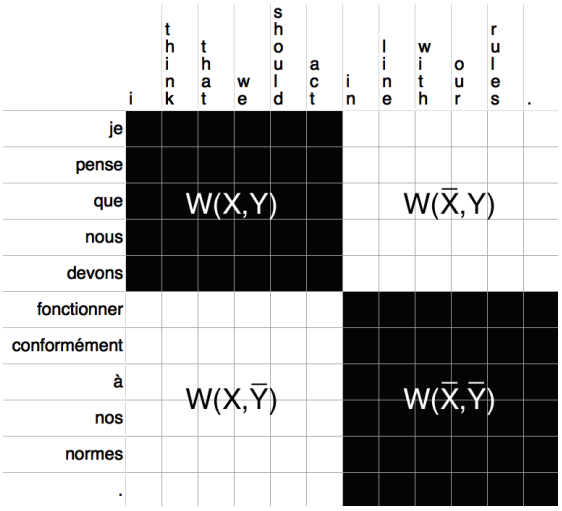

Fig. 3. Illustration of the segmentation of sentences $S = X.\overline{X}$ and $T = Y.\overline{Y}$. Here the block we start with is the entire matrix. Splitting horizontally and vertically into two parts gives four sub-blocks. There are two possible directions of segmentation: *linear* with the two sub-blocks in black, or *cross* with the two sub-blocks in white. The process is repeated recursively in the selected direction.

Table 3 reports the ratio of difference in final segmentation between the original approach and our approach on 350,000 French-English sentence pairs. It is only 0.3% in total. Differences start to appear only after the third level of segmentation and occur only once in 10,000 cases on that level. These figures show that the use of cut, instead of Ncut, for the decision on the direction does not significantly affect the final segmentation results. cut is enough to determine direction, while the introduction of its variant, Ncut, is justified to select the most dense pairs of sub-blocks, so as to lead to better alignments. As for time, a reduction of around 1/3 of computation time is observed. This will also be visible in Table 6 where the versions of Cutnalign denoted A use cut instead of Ncut for the decision on direction.

Figure 2 visualised a sentence pair matrix before sub-sentential alignment. Following intuition, the higher the translation strength between two words, the more they are prone to participate in the final sub-sentential alignment. Figure 4 shows this. Experiments on 347,614 French–English sentence pair matrices. showed, that, in average, in each sentence, less than 3% of the word pairs have a translation strength higher than 0.1. More than 75% of these word pairs belong to the final sub-sentential alignment. We will now exploit this trend to reduce the search space in a sentence pair matrix.

Table 3. Percentage of segmentation differences when using cut instead of Ncut to decide for direction, on 347,614 French–English sentence pairs. The first column is the level of segmentation. The two middle columns give the average length of sub-blocks in source and target at that level. The last column gives the percentage of differences in the final segmentation when using approximate calculation; the reference is the final segmentation obtained using the original method.

Segmentation level	Avg length of block (in words)		Differences (%)
	in source	in target	
0	31	28	0.00
1	21	19	0.00
2	15	14	0.00
3	11	10	0.01
4	8	7	0.02
5	7	6	0.04
6	6	5	0.07
7	5	5	0.09
>7	4	3	0.11

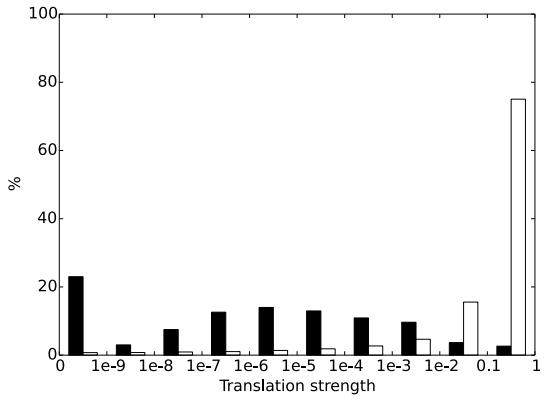

Fig. 4. Percentage of cells in the matrix with a translation strength inside the bin (*black bars*) and percentage of word pairs actually participating in the final sub-sentential alignment (*white bars*). The bars are drawn for each logarithmic bin on the horizontal axis: $[0, 10^{-9})$, $[10^{-9}, 10^{-8})$, ... $[0.1, 1]$.

3.2 Reduction of the Search Space

So as to decide the direction and the location for splitting into two sub-blocks, cuts are computed at *each* point inside a block. We propose to compute a kind of mask on the sentence pair matrix so as to restrict in advance the choice for splitting points at any level during segmentation.

In a preprocessing phase, all cells with a translation strength higher than a threshold are identified. We call them *peak cells*. As an illustration, consider the 5 black cells with a translation strength higher than 0.1 in the top matrix on the left of Fig. 5 (same as Fig. 2): (je, i), (nous, we), (nous, our), (nos, our) and (., .).

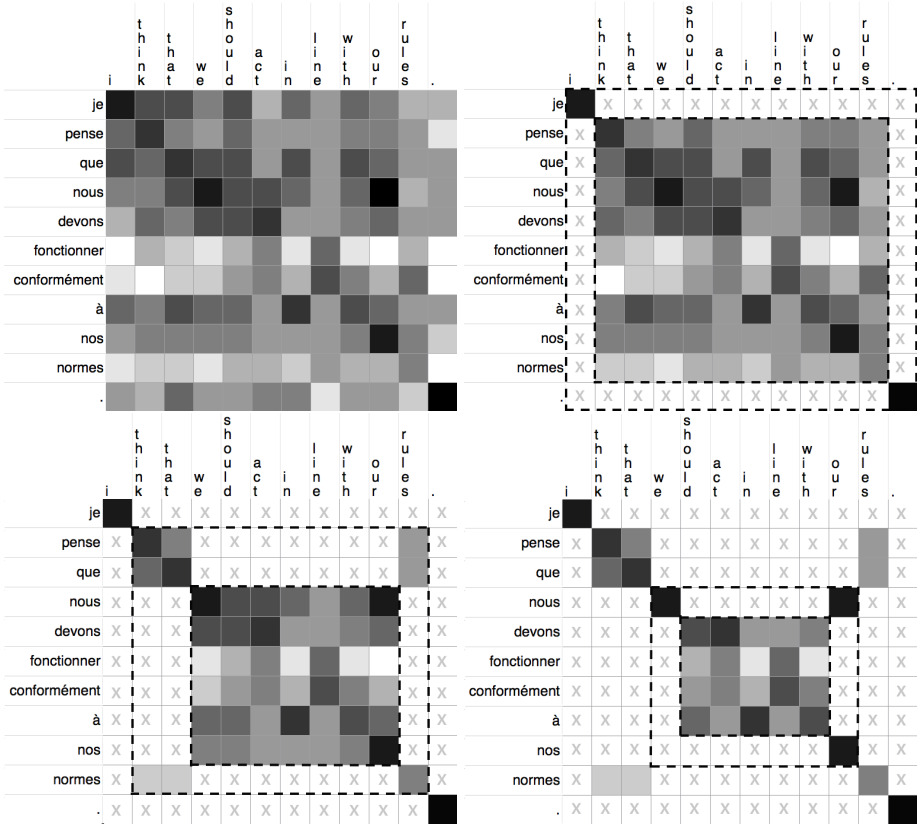

Fig. 5. Reduction of the search space in a French–English sentence pair matrix (same as Fig. 2) to find the sub-sentential alignment. Cells are grayed from 0.0 (white) to 1.0 (black) on a logarithmic scale to visualise translation strengths.

The next phase, the reduction phase, processes the matrix step by step. At the beginning of the first step, the domain is the entire sentence pair matrix and the search space is empty. In each step, the following operations are performed.

Firstly, the smallest rectangle with the largest number of peak cells is determined. The reason to select such a rectangle follows the intuition behind the introduction of Ncut: by approximation, the smallest rectangle with the largest number of peak cells should lead to the extraction of the densest sub-blocks. Necessarily, such a rectangle is delimited by some peak cells, which are added to the search space. The top matrix on the right in Fig. 5 shows the rectangle obtained in the first iteration step. It is the outer rectangle visualized by dotted lines. It is delimited by the peak cells (je, i) and (., .). The bottom matrix shows the one obtained in the second iteration step (outer rectangle again). It is delimited by the peak cells: (nous, we), (nous, our) and (nos, our). In all generality, peak cells do not necessarily lie in the corners; Figure 5 is a particular case.

Then, the next domain for the next step of iteration is determined by leaving out the cells in the contour which are not peak cells. In the two matrices on the right of Fig. 5, such new domains are the *inner* rectangles delimited by dotted lines. This leaves out the cells containing a grey cross in the figure. This can be done with some confidence because, by construction, sub-blocks extracted from such locations will leave out many well aligned word pairs and cannot be expected to yield a promising Ncut. On the contrary, one can expect that sub-blocks determined by splitting on positions in the new domain will be denser in well aligned word pairs.

Finally, the corner regions between two successive smallest rectangles are added to the search space (see bottom left matrix in Fig. 5), because the positions inside these regions have a good chance to provide a higher number of well aligned word pairs when splitting into sub-blocks.

The new domain is passed to the next iteration step. The iteration process stops when the smallest rectangle contains one or zero peak cell. In this case, the search space is not reduced and used as is.

The final reduced search space is thus made out of all the peak cells, all the corner regions between two successive smallest rectangles and the last inner rectangle. It usually takes the rough shape of a cross extending over the sentence pair matrix. This reduced search space is then passed over to the general sub-sentential alignment process which will no more be allowed to consider any possible positions to split into sub-blocks, but will be confined to the positions in the reduced search space at any level. As a consequence, processing time will be reduced: for well-balanced cases, a reduction from a computation in $O(n^2)$ to $O(n \log n)$ is obtained. The experiments reported hereafter also show that the reduction in search space does not affect BLEU scores.

The procedure described above for reduction of space was first implemented in Python. Its re-implementation in C divides the processing time by 6 (see Table 6, last line).

4 Experiments

4.1 Overview: Tools and Data

We evaluate our work by building PB-SMT systems using the Moses toolkit [11], lexicalised reordering models [10] and the KenLM Language Modelling toolkit [8]. Accuracy relatively to translation references is assessed using BLEU. Baseline systems are built using GIZA++ or Fast align. For systems built using Anymalign and Cutnalign, the overall process to build translation tables has been illustrated in Fig. 1.

All the experiments mentioned in this paper use data from corresponding parts of the Europarl parallel corpus v3 [9], so that BLEU scores can be compared across language pairs, as the training, tuning and test sets correspond across languages. The training corpus is made of 347,614 sentences; 500 sentences are used for tuning; the test set contains 5,000 lines.

We use 3 language pairs in both directions involving 5 European languages: English (en), French (fr), Spanish (es), Portuguese (pt), Finnish (fi): fr–en as usual test languages, fi–en, i.e., agglutinative language vs. isolating language, and es–pt, as an example of close languages. Statistics on the data are given in Table 4.

Table 4. Statistics on the data used (M = million)

		en	fr	es	pt	fi
Train	sentences			347,614		
	word tokens	10.01M	11.62M	10,52M	10.35M	7.21M
	word types	57,728	72,042	124,035	116,165	289,054
Tune	sentences			500		
	word tokens	14,697	17,132	15,440	15,348	10,580
	word types	2,929	3,395	3,489	3,595	4,576
Test	sentences			5,000		

4.2 Timeout and Max Length of Phrases for Anymalign

As Anymalign should be given a timeout, we first inquire the relationship between the timeout and translation accuracy. In addition, we also inquire the influence of the max length of the phrases output by Anymalign on translation accuracy.

Table 5 shows that 15 minutes is enough to get comparable translation accuracy. For all language pairs and timeouts, almost all the best scores are obtained for a max length of phrases set to 2. Drawing from these results, the experiments

Table 5. Translation accuracy (BLEU) against timeout and maximal length of phrases output by Anymalign for three language pairs. Here the test set contains 38,000 lines.

Language pair	Max length of phrases	BLEU for ≠ timeouts 15 min	30 min	60 min	120 min
fr–en	1	33.9	34.1	34.0	34.1
	2	34.0	34.3	34.1	34.2
	3	34.0	33.7	34.4	34.3
	4	33.7	33.8	34.2	34.1
pt–es	1	38.5	38.6	38.6	38.6
	2	38.5	38.7	38.7	38.8
	3	38.5	38.4	38.8	38.9
	4	38.4	38.5	38.6	38.8
fi–en	1	23.0	23.3	23.6	23.8
	2	23.3	23.2	23.8	24.2
	3	22.8	23.4	23.5	23.7
	4	22.7	23.2	23.6	23.9

reported hereafter will adopt the following settings: Anymalign will be run with a max length of phrases set to 2. This will be denoted by Anymalign -i2. The original version of Anymalign will be given a time-out of 15 minutes, while its multi-processor version will be given a time-out of 5 minutes, according to the acceleration reported in Sec. 2.2.

4.3 Incremental Improvements

We incrementally evaluated the improvements presented in the previous sections on French–English data. In order to evaluate the difference in the final sub-sentential alignments obtained, we measured the alignment error rate (AER) [1] by reference to the results obtained using the original methods without any improvement. As seen in Table 6, the multi-processing implementation of Anymalign delivers more word alignments than the mono-processor implementation in 3 times less time. In total, we could divide the training time by 750 without affecting the BLEU scores. Differences in alignements are observed but positively impact the results.

4.4 Comparison with Fast align

We compare the integration of all improvements with the fastest probabilistic state-of-the-art alignment method: Fast align. We run it with default settings in two directions, source to target and target to source, to produce alignments from which a phrase table is extracted using the grow-diag-final-and heuristic.

Table 6. Incremental gains in time on French–English data. The max length of phrases output by Anymalign is set to 2 in all experiments. M denotes a multi-processing version (number of cores used: 4). For Cutnalign, S avoids unnecessary subtraction of zeros; A uses cut instead of Ncut to make the decision on direction of segmentation (Sect. 3.1); R implements reduction of search space (Sect. 3.2, threshold for translation strength set to 0.5); C uses re-implementation in C of core component of Cutnalign. The alignment time is the time for Anymalign plus the time for Cutnalign. In total a speed-up by 750 has been obtained (4,515 / 6).

Anymalign -i2 + Cutnalign	Alignment time (min)	AER (%)	BLEU (%)
original + original	15 + 4,500	–	34.0 ± 0.8
original + M	15 + 1,594	0.0	34.0 ± 0.8
original + M+S	15 + 31	0.0	34.0 ± 0.8
M + M+S	5 + 32	0.0	34.1 ± 0.8
M + M+S+A	5 + 19	5.4	34.2 ± 0.8
M + M+S+R	5 + 15	8.8	34.0 ± 0.8
M + M+S+A+R	5 + 6	11.8	34.1 ± 0.8
M + M+S+A+R+C	5 + 1	11.8	34.1 ± 0.8

For Anymalign, we use the options -i 2 -t 300, i.e., we set a preferred length of up to 2 words in associations, and a timeout of 5 minutes.

The results of the experiments are presented in Table 7. Our improvements allow the associative methods to beat Fast align in time. In addition, as much smaller phrase tables are extracted by our method, lower times for decoding are observed. Alignments produced with our improvements yield slightly lower scores than those obtained with Fast align on French–English and Spanish–Portuguese in both directions, but with no statistically significant difference in each case as the confidence intervals show. Unfortunately, on Finnish–English, in both directions, our BLEU scores are significantly lower. This may come from an insufficient timeout for Anymalign, 5 minutes, chosen for the sake of consistency across all experiments reported in this section.

5 Conclusion

We presented multi-processing implementations of the multilingual sampling-based alignment method [14] and of the hierarchical sub-sentential alignment method [13], two associative methods which, by essence allow for this. We introduced two approximations in the hierarchical sub-sentential alignment method: we modified how to decide the direction of split and we reduced the search space. The removal of some unnecessary computations, and the re-implementation of core components in C were also introduced. We obtained considerable gains in time so that the combination of these two associative methods becomes competitive with probabilistic methods.

The new multi-processing version of Anymalign divides the computation times for word associations by 3 on a 4-core computer. Elimination of unneces-

Table 7. Comparison of BLEU scores and alignment times in 6 language pairs with different aligners

Language pair	Aligner	Align. time (min)	BLEU (%)
pt-es	MGIZA++	150	39.1 ± 0.8
	Fast align	17	38.9 ± 0.8
	M + M+S+A+R+C	**7**	38.8 ± 0.8
es-pt	MGIZA++	140	37.1 ± 0.8
	Fast align	17	36.9 ± 0.8
	M + M+S+A+R+C	**7**	36.6 ± 0.8
en-fr	MGIZA++	150	36.3 ± 0.7
	Fast align	17	36.1 ± 0.7
	M + M+S+A+R+C	**7**	36.0 ± 0.7
fr-en	MGIZA++	170	34.5 ± 0.8
	Fast align	17	34.5 ± 0.8
	M + M+S+A+R+C	**7**	34.1 ± 0.8
fi-en	MGIZA++	120	26.1 ± 0.8
	Fast align	14	25.0 ± 0.8
	M + M+S+A+R+C	**6**	23.9 ± 0.8
en-fi	MGIZA++	110	16.3 ± 0.8
	Fast align	14	16.7 ± 0.8
	M + M+S+A+R+C	**6**	15.7 ± 0.8

sary computations of special cases in Cutnalign divides the computation times of sub-sentential alignments by more than 40 times in comparison with the original implementation described in [13]. Combined with multi-processing, this leads to a speed-up of roughly 140 times on a 4-core computer. The latest version of Cutnalign, which also includes approximations in decisions and reduction of the search space, and C implementation of core components runs approximately 4,500 times faster than the original implementation. The combination of the two new versions of Anymalign and Cutnalign result in an overall alignment process that can be faster than Fast align while delivering comparable results.

As for comparison with probabilistic methods, some may argue that the comparison of a probabilistic method running on one processor with an associative method running on 4 cores is unfair. We claim on the contrary that it *is* fair because associative methods intrinsically cater for this at no expense of the quality of their results. What would be unfair is precisely to forbid associative methods to make use of their inherent advantages.

Our main task in the near future is to get rid of the threshold manually set for the reduction of the search space in the hierarchical sub-sentential alignment method. We want to find a way to automatically determine a threshold based on an inspection of the distribution of translation strengths in sentence pair matrices.

As a final note, it is worth mentioning that both Anymalign and Cutnalign are by essence bidirectional methods. They compute the bidirectional parameters

or the sub-sentential alignments in one pass, on the contrary to MGIZA++ or Fast align which have to be run in both directions and on the contrary to the first step in grow-dial-final-and which builds two matrices in both directions before merging.

References

1. Ayan, N.F., Dorr, B.J.: Going beyond AER: An extensive analysis of word alignments and their impact on MT. In: Proc. of COLING/ACL. pp. 9–16 (2006)
2. Brown, P.F., Pietra, V.J.D., Pietra, S.A.D., Mercer, R.L.: The mathematics of statistical machine translation: Parameter estimation. Computational linguistics 19(2), 263–311 (1993)
3. Dunning, T.: Accurate methods for the statistics of surprise and coincidence. Computational linguistics 19(1), 61–74 (1993)
4. Dyer, C., Chahuneau, V., Smith, N.A.: A simple, fast, and effective reparameterization of IBM model 2. In: Proc. of HLT-NAACL. pp. 644–648 (2013)
5. Gale, W.A., Church, K.W.: Identifying word correspondences in parallel texts. In: Proc. of the workshop on Speech and Natural Language. vol. 91, pp. 152–157 (1991)
6. Gao, Q., Vogel, S.: Parallel implementations of word alignment tool. In: Software Engineering, Testing, and Quality Assurance for Natural Language Processing. pp. 49–57 (2008)
7. Gong, L., Max, A., Yvon, F.: Improving bilingual sub-sentential alignment by sampling-based transpotting. In: Proc. of IWSLT. pp. 243–250 (2013)
8. Heafield, K.: Kenlm: Faster and smaller language model queries. In: Proc. of the 6th Workshop on Statistical Machine Translation. pp. 187–197 (2011)
9. Koehn, P.: Europarl: A parallel corpus for statistical machine translation. In: Proc. of Machine Translation Summit. vol. 5, pp. 79–86 (2005)
10. Koehn, P., Axelrod, A., Birch, A., Callison-Burch, C., Osborne, M., Talbot, D., White, M.: Edinburgh system description for the 2005 IWSLT speech translation evaluation. In: Proc. of IWSLT. pp. 68–75 (2005)
11. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R.: Moses: Open source toolkit for statistical machine translation. In: Proc. of ACL (Poster sessions). pp. 177–180 (2007)
12. Lardilleux, A., Yvon, F., Lepage, Y.: Hierarchical sub-sentential alignment with Anymalign. In: Proc. of EAMT 2012. pp. 279–286 (2012)
13. Lardilleux, A., Yvon, F., Lepage, Y.: Generalizing sampling-based multilingual alignment. Machine translation 27(1), 1–23 (2013)
14. Levenberg, A., Callison-Burch, C., Osborne, M.: Stream-based translation models for statistical machine translation. In: Proc. of HLT-NAACL. pp. 394–402 (2010)
15. Och, F.J., Ney, H.: A systematic comparison of various statistical alignment models. Computational linguistics 29(1), 19–51 (2003)
16. Smaïli, K., Jamoussi, S., Langlois, D., Haton, J.P.: Statistical feature language model. In: Proc. of ICSLP. pp. 1357–1360 (2004)
17. Zha, H., He, X., Ding, C., Simon, H., Gu, M.: Bipartite graph partitioning and data clustering. In: Proc. of int. conf. on Info. and Knowledge Management. pp. 25–32 (2001)