# Multi-processing and approximations in associative methods for faster training of PB-SMT systems

**Baosong Yang*** **and Yves Lepage***

* Graduate School of Information, Production and Sytems, Waseda University
Hibikino 2-7, Wakamatsu-ku, Kitakyushu-shi, 808-0135 Fukuoka-ken, Japan
yangbaosong@fuji.waseda.jp, yves.lepage@waseda.jp

**Abstract**

Training statistical machine translation systems used to require heavy computation times. Recent work (`Fast align`) achieved impressing improvements in the probabilistic approach. We show that, by leveraging the advantages of the associative approach, we achieve similar, even faster, training times, while keeping comparable BLEU scores.

## 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 (Brown et al., 1993) and the grow-diag-final-and heuristic are the most popular approach. They have been integrated as the `GIZA++` tool (Och and Ney, 2003), or `MGIZA++` (Gao and Vogel, 2008) for a parallel implementation, in the PB-SMT toolkit Moses[1]. A log-linear re-parameterization of IBM Model 2 has recently been implemented in `Fast align`[2] (Dyer et al., 2013) and leads to much faster training times.

IBM models are probabilistic models, so that the optimization process requires the knowledge of the entire parallel corpus to estimate the parameters (Levenberg et al., 2010). On the contrary, *associative* methods, as characterized in (Gale and Church, 1991), do not rely on a global alignment model, but use local maximization so that each sentence pair can be processed independently.

*Sampling-based multilingual alignment*, introduced in (Lardilleux et al., 2013), 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 (Lardilleux et al., 2012), 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 (Gong et al., 2013).

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.

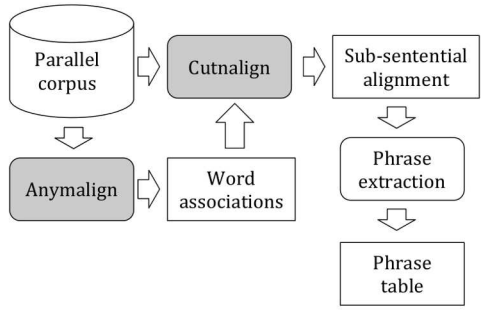


Figure 1: Combination of two associative methods, `Anymalign` and `Cutnalign`, to obtain phrase tables from a parallel corpus.

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. 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.

## 2. Multi-processing

### 2.1. 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. 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

[1] `http://www.statmt.org`
[2] `http://github.com/clab/fast_align`
[3] `https://anymalign.limsi.fr/`
[4] Thanks to the authors for providing the source code.
[5] `train-model.perl --first step 4`

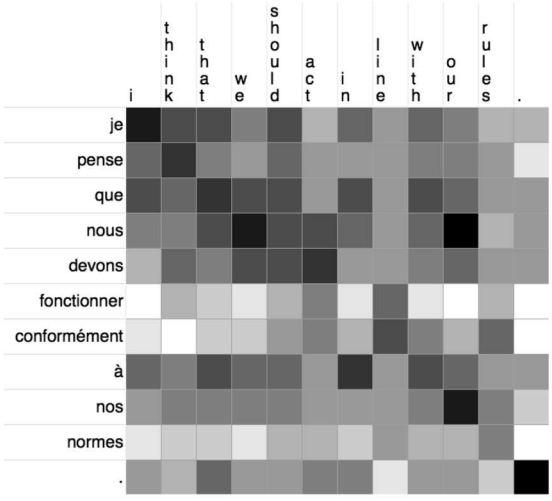


Figure 2: Translation strengths in a French–English sentence pair matrix. Cells are grayed from 0.0 (white) to 1.0 (black) on a logarithmic scale.

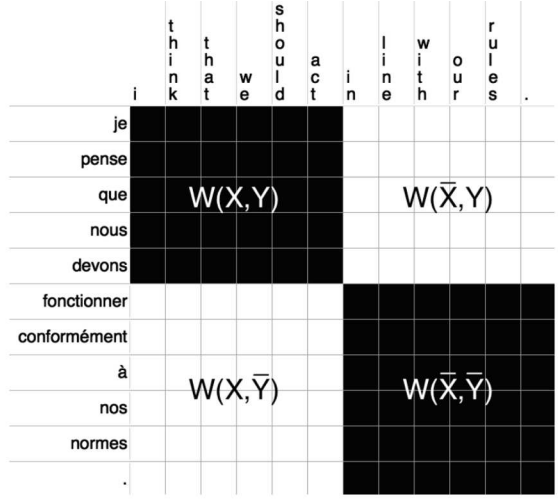


Figure 3: Illustration of the segmentation of sentences $S = X.\overline{X}$ and $T = Y.\overline{Y}$. Here the block we start with is the entire matrix. Splitting horizontally and vertically into two parts gives four sub-blocks. There are two possible directions of segmentation: *linear* with the two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

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.

### 2.2. Hierarchical sub-sentential alignment

`Cutnalign` is easily parallelized by observing that the sub-sentential alignment process for each different sentence pair is independent from the other ones. 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 parallelized and non-parallelized implementations are theoretically equivalent. We checked that sub-sentential alignments outputs in both implementations are exactly the same.

## 3. Two approximations in hierarchical sub-sentential alignment

The original sub-sentential alignment method proposed in (Lardilleux et al., 2012) 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 sentence pair matrix is bi-directional by construction.

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

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 (Zha et al., 2001) for document clustering. For this purpose, a score named cut (see Equation 1) is computed that sums up[7] all the cells in the two sub-blocks of a block in the sentence pair matrix (see Figure 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 $(X,Y)$.

$$\mathrm{cut}(X,Y) = W(X,\overline{Y}) + W(\overline{X},Y) \qquad (1)$$

In order to make the partition as dense as possible, Zha et al., 2001) use a normalized variant of the score named Ncut (see Equation 2)[8]. The best segmentation minimizes this variant over all $\mathrm{Ncut}(X,Y)$ and $\mathrm{Ncut}(\bar{X},Y)$, thus making simultaneously the decision of where to split and in which direction.

$$\mathrm{Ncut}(X,Y) = \frac{\mathrm{cut}(X,Y)}{\mathrm{cut}(X,Y) + 2{\times}W(X,Y)} + \frac{cut(\bar{X},\bar{Y})}{\mathrm{cut}(\bar{X},\bar{Y}) + 2{\times}W(\bar{X},\bar{Y})} \qquad (2)$$

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

[7] A first engineering improvement over the original implementation, that leads to a large speed-up, was to avoid unnecessary subtractions of zeros that were introduced in the original code by using an elegant, but inefficient, formula for summations. This improvement is noted by the abbreviation S in Table 1.

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

### 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 Figure 3, and
- which direction, linear or cross, i.e., choosing either the black segmentation or the white one in Figure 3.

The original approach consists in making the two decisions simultaneously, by selecting the max over all possible $\mathrm{Ncut}(X, Y)$ and $\mathrm{Ncut}(\bar{X}, Y)$. 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.

We measured 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. As for time, a reduction of around 1/3 of computation time is observed. This is visible in Table 1 where the versions of `Cutnalign` denoted A use cut instead of Ncut for the decision on direction.

Figure 2 visualizes 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. Experiments on 350,000 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.

### 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

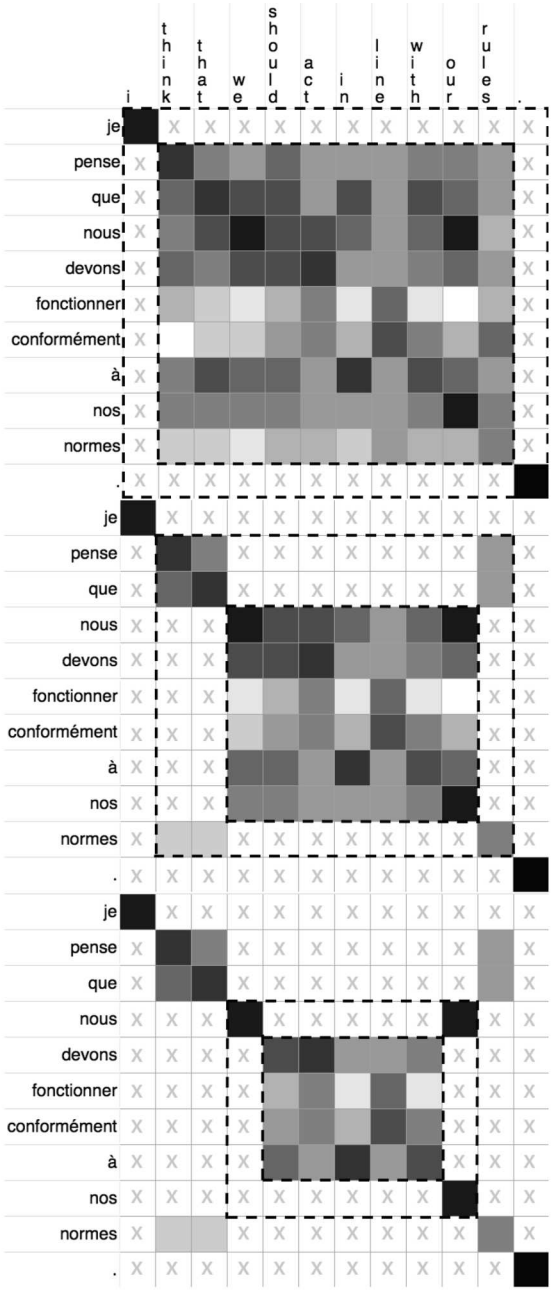


Figure 4: Reduction of the search space

cells with a translation strength higher that 0.1 in Figure 2: (je, i), (nous, we), (nous, our), (nos, our) and (., .).

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

| 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 |

Table 1: Incremental gains in time on French–English data. M denotes a multi-processing version (number of cores used: 4). For Cutnalign, S avoids unnecessary subtraction of zeros (Footnote 7); A uses $cut$ instead of $Ncut$ to make the decision on direction of segmentation (Section 3.1.); R implements reduction of search space (Section 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)

space. The top matrix in Figure 4 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 4 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 top and bottom matrices of Figure 4, 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 middle matrix in Figure 4), 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. 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[9]. 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 1, last line).

## 4. Experiments

We evaluate our work by building PB-SMT systems using the Moses toolkit, lexicalized reordering models (Koehn et al., 2005) and the KenLM Language Modeling toolkit (Heafield, 2011). Accuracy relatively to translation references is assessed using BLEU.

All the experiments mentioned in this paper use the data from the corresponding part of the Europarl parallel corpus v3 (Koehn, 2005), 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.

### 4.1. 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) (Ayan and Dorr, 2006) by reference to the results obtained using the original methods without any improvement.

As seen in Table 1, we could divide the training time by 750 without affecting the BLEU scores. Differences in alignements are observed but positively impact the results.

### 4.2. 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. 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.

We use 3 language pairs in both directions involving 5 European languages[10]: fr–en (usual test languages), fi–

[9]For well-balanced cases, a reduction from a computation in $O(n^2)$ to $O(n \log n)$ is obtained.

[10]English (en), French (fr), Spanish (es), Portuguese (pt), Finnish (fi).

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

Table 2: Comparison of BLEU scores and alignment times in 6 language pairs with different aligners

en (agglutinative language–isolating language), and es–pt (close languages).

The results of the experiments are presented in Table 2. Our improvements allow the associative methods to beat `Fast align` in time. 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 consistency across all experiments reported here.

## 5. Conclusion

We presented multi-processing implementations of the multilingual sampling-based alignment method (Lardilleux et al., 2013) and of the hierarchical sub-sentential alignment method (Lardilleux et al., 2012), 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.

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.

## 6. Acknowledgements

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 her training period in Waseda University.

## 7. References

Ayan, Necip Fazil and Bonnie J. Dorr, 2006. Going beyond AER: An extensive analysis of word alignments and their impact on MT. In *Proc. of COLING/ACL*.

Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer, 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Dyer, Chris, Victor Chahuneau, and Noah A. Smith, 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proc. of HLT-NAACL*.

Gale, William A. and Kenneth Ward Church, 1991. Identifying word correspondences in parallel texts. In *Proc. of the workshop on Speech and Natural Language*, volume 91.

Gao, Qin and Stephan Vogel, 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*.

Gong, Li, Aurélien Max, and François Yvon, 2013. Improving bilingual sub-sentential alignment by sampling-based transpotting. In *Proc. of IWSLT*.

Heafield, Kenneth, 2011. Kenlm: Faster and smaller language model queries. In *Proc. of the 6th Workshop on Statistical Machine Translation*.

Koehn, Philipp, 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of Machine Translation Summit*, volume 5.

Koehn, Philipp, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, David Talbot, and Michael White, 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proc. of IWSLT*.

Lardilleux, Adrien, François Yvon, and Yves Lepage, 2012. Hierarchical sub-sentential alignment with Anymalign. In *Proc. of EAMT 2012*.

Lardilleux, Adrien, François Yvon, and Yves Lepage, 2013. Generalizing sampling-based multilingual alignment. *Machine translation*, 27(1):1–23.

Levenberg, Abby, Chris Callison-Burch, and Miles Osborne, 2010. Stream-based translation models for statistical machine translation. In *Proc. of HLT-NAACL*.

Och, Franz Josef and Hermann Ney, 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Zha, Hongyuan, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu, 2001. Bipartite graph partitioning and data clustering. In *Proc. of int. conf. on Info. and Knowledge Management*.