Building a Large Machine-Aligned Parallel Treebank

Jörg Tiedemann
Department of Linguistics and Philology
Uppsala University, Uppsala/Sweden
jorg.tiedemann@lingfil.uu.se

Gideon Kotzé
Alpha-Informatica
Rijksuniversiteit Groningen, Groningen, The Netherlands
g.j.kotze@rug.nl

Abstract

This paper reports on-going work on building a large automatically tree-aligned parallel treebank in the context of a syntax-based machine translation (MT) approach. For this we develop a discriminative tree aligner based on a log-linear model with a rich feature set. We incorporate various language-independent and language-specific features taking advantage of existing tools and annotation. Our initial experiments on a small hand-aligned treebank show promising results even with small amounts of training data. The performance of our approach is well above unsupervised techniques reported elsewhere. This enables us to quickly create training material and alignment models for additional language pairs. In recent work, we aligned more than one million sentence pairs and started our experiments with the extraction of transfer knowledge for our example-based machine translation system.

1 Introduction

A parallel treebank consists of a collection of sentence pairs that have been grammatically tagged, syntactically annotated and aligned on sub-sentential level [15]. This alignment usually includes more than the alignment of words but also links between all the corresponding constituents of given sentence pairs. Parallel treebanks are valuable resources for a wide variety of applications. They are not only useful as a resource for the induction of translation knowledge for machine translation systems but also as a source for cross-lingual corpus-based studies, for example, in contrastive linguistics. However, our main concern is the use of parallel treebanks in the context of machine translation (MT). Since well-aligned treebanks will play a substantial role in our MT model, finding an optimal solution to the problem of tree alignment is very important. Various projects have been initiated to create

aligned parallel treebanks [7, 1, 5] and most of them are based on tedious manual labor. There are some approaches to automatic tree alignment facilitating syntax-oriented translation models [16, 4, 6, 17, 10]. In contrast to these unsupervised or hand-crafted alignment systems, we propose a classifier-based alignment approach based on supervised learning.

In the following section we outline our model including a discussion of features used in our experiments. Thereafter, we show our results on the development corpus and, finally, describe on-going work of applying our model to large scale data sets.

2 Discriminative Tree Alignment

In our approach we look at the alignment of nodes in phrase-structure tree pairs. We model this alignment using a discriminative feature-based model, which has the advantage that we do not have to "invent" a generative story to explain the alignment. Hence, we use a binary classifier-based approach using a probabilistic model directly predicting the likelihood of a link a_{ij} between two nodes s_i and t_j given the features associated with these nodes. For simplicity we leave the structural nature of the tree alignment problem aside for now. We will later discuss issues of structured prediction and how we address link dependencies in our approach. In the next section, we introduce the general model applied in the tree aligner.

2.1 A Log-Linear Link Prediction Model

Similar to related work on discriminative word alignment we base our model on association features extracted for each possible alignment candidate. For tree alignment, each pair of nodes $\langle s_i, t_j \rangle$ from the source and the target language parse tree is considered and a score x_{ij} is computed that represents the degree to which both nodes should be aligned according to their features $f_k(s_i, t_j, a_{ij})$ and corresponding weights λ_k derived from training data. In our approach we use conditional likelihood using a log-linear model for estimating these values:

$$P(a_{ij}|s_i,t_j) = \frac{1}{Z(s_i,t_j)} exp\left(\sum_k \lambda_k f_k(s_i,t_j,a_{ij})\right)$$

Here, the mapping of data points to features is user provided (see section 2.3) and the corresponding weights are learned from aligned training data. As mentioned earlier, we simplify the problem by predicting individual alignment points for each candidate pair instead of aiming at structured approaches. Hence, we can train our conditional model as a standard binary classification problem. Note that contextual features can easily be integrated even though first-order dependencies on surrounding alignments are not explicitly part of the model. More details will be given below in sections 2.4 and 2.6.

In our experiments we will use a standard maximum entropy classifier using the log-linear model as stated above. One of the advantages of maximum entropy classifiers is the flexibility of choosing features. No independence assumptions have to be made and state-of-the art toolboxes are available with efficient learning strategies. In this work, we apply the freely available toolbox Megam [2].

2.2 Structured Prediction

Tree alignment is a typical structured prediction problem. Treating links in isolation as outlined above causes a lot of errors due to the existing dependencies between link decisions. Therefore, applying a binary classifier alone and linking according to the individual decisions is not a good idea. However, building a discriminative predictor for the entire structure is not feasible due to the sparsity of the data. Structured prediction is an active research area and various approaches have been proposed incorporating various dependencies and standard machine learning techniques. In our implementation we opt for a *recurrent architecture* [3] using history-based features and a sequential classification process. We experimented with two different directions: top-down and bottom-up link classification where the latter gave us better results. Therefore, we will only report these results.

The idea behind this strategy is simple: previous decisions of the global classifier are used as input features for coming decisions. Training is simple as link decisions are readily available and the classifier can learn directly using those features. In classification, we have to use a sequential setup in order to obtain history features. In the bottom-up strategy we start with predicting links for leaf nodes moving up towards the tree root. Here, we assign history features to be taken from the child nodes.

Another way of incorporating structural dependencies in prediction is to use a simple greedy alignment strategy. In tree alignment, it is common to restrict the process to one link per node. Therefore, we can define a greedy best-first alignment strategy to account for competition between link candidates [12]. Further constraints can be applied to guide the alignment even further. For example, well-formedness criteria can be defined to reject certain links [17]. These constraints basically add the following restriction: Descendants/ancestors of a source linked node may only be linked to descendants/ancestors of its target linked counterpart. We apply both, a greedy best-first strategy and well-formedness constraints. In addition, we also introduce a constraint to restrict alignments in such a way that non-terminal nodes are aligned to non-terminal nodes and terminal nodes to terminal nodes only.

2.3 Basic Alignment Features

Log-linear models are very flexible with regard to the feature functions that can be used. Any real-valued function can be used without considering their dependencies with each other.

Zhechev & Way [17] introduce lexical probabilities to be used in unsupervised tree alignment. They are combined into an alignment score γ which is composed out of so-called *inside* scores ($\alpha(s_l|t_l)$, $\alpha(t_l|s_l)$,) and *outside* scores ($\alpha(\overline{s_l}|\overline{t_l})$, $\alpha(\overline{t_l}|\overline{s_l})$). We use a slightly modified definition of inside/outside scores which involves the selection of the maximum lexical score for each token instead of an averaged sum over all possible word connections ($x_i \geq x$ denoting dominance):

$$\begin{aligned}\gamma(s_l,t_l) &= \alpha(s_l|t_l)\alpha(t_l|s_l)\alpha(\overline{s_l}|\overline{t_l})\alpha(\overline{t_l}|\overline{s_l}) \\ \alpha(x|y) &= \prod_{x_i \geq x} max_j P(x_i|y_j)\end{aligned}$$

In our experiments this modification gave us a better performance and intuitively this is also more appealing. Other features can be derived directly from an existing word alignment. We define a feature measuring the proportion of *consistent* links among all *relevant* links L_{xy} involving either source s_x or target language words t_y dominated by the current tree nodes (s_i and t_j).

$$\begin{aligned}align(s_i,t_j) &= \sum_{L_{xy}} consistent(L_{xy},s_i,t_j) / \sum_{L_{xy}} relevant(L_{xy},s_i,t_j) \\ consistent(L_{xy},s_i,t_j) &= \begin{cases} 1 & \text{if } s_x \geq s_i \wedge t_y \geq t_j \\ 0 & \text{otherwise} \end{cases} \\ relevant(L_{xy},s_i,t_j) &= \begin{cases} 1 & \text{if } s_x \geq s_i \vee t_y \geq t_j \\ 0 & \text{otherwise} \end{cases}\end{aligned}$$

Any sub-sentential alignment can be used with the definition above. In our experiments we apply the Viterbi word alignments produced by Giza++ [13] using the IBM 4 model in both directions, the union of these links and their intersection. This gives us four separate feature functions using the definition above.

Another word alignment feature is defined for pairs of terminal nodes. It is simply a binary feature being set to one if and only if both nodes are linked in the underlying word alignment. This is useful if terminal node alignment is included in the tree alignment model as it is in our initial experiments.

It is also possible to define a number of features that are independent of external tools and resources. Using the relative position of each node in the parse tree we define two features: tree-level similarity (tls) and tree span similarity (tss). For the former we use the distance to the root node ($d(s_i,s_{root})$ resp. $d(t_i,t_{root})$) and normalize this distance with the size of the tree (maximum distance between any node and the root node). For the latter we compute the relative "horizontal" position of a node based on the span ($s_{start} \rightarrow s_{end}$) of the entire subtree which is rooted in that node (s_i). This position is than normalized by the length (= number of leaf nodes) of the sentence (S). Furthermore, we define a *leafratio* feature that measures the difference in subtree spans (in terms of number of leaf nodes within that subtree). The formal definitions of these *tree features* are as follows:

$$tls(s_i,t_j) = 1 - abs\left(\frac{d(s_i,s_{root})}{max_x d(s_x,s_{root})} - \frac{d(t_i,t_{root})}{max_x d(t_x,t_{root})}\right)$$

$$tss(s_i,t_j) = 1 - abs\left(\frac{s_{start}+s_{end}}{2*length(S)} - \frac{t_{start}+t_{end}}{2*length(T)}\right)$$

$$leafratio(s_i,t_j) = \frac{min(|leafnodes(s_i)|,|leafnodes(t_j)|)}{max(|leafnodes(s_i)|,|leafnodes(t_j)|)}$$

Finally, we also define features derived from the annotation. Intuitively, category labels (for non-terminal nodes) and part-of-speech labels should be valuable indicators for a possible link. These features are simply binary features which are set to one if the particular label combination is present and zero otherwise.

2.4 Contextual Features

So far we only considered features directly attached to the candidate nodes. However, tree nodes are connected with other nodes in the tree structure and their alignment may as well depend on features of neighboring nodes. Therefore, features from surrounding nodes should be considered as well. Using the tree structure we can extract the same features as described above from other nodes connected to candidate nodes. For this we define the following functions that can be used to move within the tree when extracting features: *parent* - move to the immediate parent and take the feature values from this node; *child* - compute the average feature value for all child nodes; *sister* - compute the average feature value for all nodes with the same parent node. These functions can be applied recursively. For example, applying *parent* twice will force the feature extraction process to move to the grand-parent node (if this node exists). Note that these functions can be applied to either source or target language tree or both. In this way we have many possibilities to explore contextual features. Proper feature engineering is necessary to define useful templates.

2.5 Complex Features

Some features may be correlated in a non-linear way. To account for those it is possible to create complex features. Any of the features above (also contextual ones) can be combined in such a way that they form a new feature function with their values combined. In our approach we simply compute the product of the feature values. Other types of combinations might be possible as well. This gives us a combinatorial explosion of possible features and careful feature engineering is necessary again to select valuable ones. Furthermore, complex features are even more exposed to sparseness problems. Nevertheless, various combinations lead to significant improvements as we will see in our experiments.

2.6 Link Dependency Features

The last category of features refers to history features mentioned earlier in our discussion on structural prediction. In our implementation we use two types of history features: (1) The *children_links* feature is the number of links between direct child nodes of the current node pair. (2) The *subtree_links* feature is the number of links between all children of the current node pair. Both values are normalized with the maximum number of child nodes on either source or target side. In classification, only the prediction likelihood is used for estimating these feature values. In other words, we use "soft counts" instead of counting actual links. Classification can then be done in a bottom-up fashion before applying the greedy best-first search in the final step.

3 Alignment Results

During the development of our tree aligner we applied the Smultron treebank [7] for evaluating its performance with various strategies and feature sets. Smultron includes two trilingual parallel treebanks in English, Swedish and German. The corpus contains the alignment of English-Swedish and German-Swedish phrase structure trees from the first two chapters of the novel "Sophie's World" by Jostein Gaarder and from economical texts taken from three different sources. The alignment has been done manually using the Stockholm Tree Aligner [11]. The alignment includes *good* links and *fuzzy* links. We will use both but give them different weights in training (good alignments get three times the weight of fuzzy and negative examples). We mainly worked with the English-Swedish treebank of Sophie's World which includes roughly 500 sentences per language (6,671 good links and 1,141 fuzzy links). In our experiments we used the first 100 aligned parse trees for training and the remaining part for testing.

For evaluation we use the standard measures of precision, recall and balanced F-scores as they are used in word alignment evaluation. Due to the distinction between good and fuzzy alignments we compute values similar to word alignment evaluation scores in which "sure" and "possible" links are considered (S refers to the good alignments in the gold standard and P refers to the possible alignments which includes both, good and fuzzy links; A are the links proposed by the system):

$$Precision = \frac{|P \cap A|}{|A|} \qquad Recall = \frac{|S \cap A|}{|S|}$$

$$F = \frac{2 * Precision * Recall}{Precision + Recall}$$

The upper part of table 1 summarizes the results for different feature sets when running the aligner on our development corpus. The alignment results are very promising. We can see that adding features consistently helps to improve the performance. The advantage of a discriminative approach with a rich feature set can

settings	Precision	Recall	F
lexical features	64.89	51.00	57.11
+ tree features	56.80	60.70	58.68
+ alignment features	62.94	62.83	62.88
+ label features	77.20	74.44	75.79
+ context features	**79.77**	**75.66**	**77.66**
train=novel, test=economy	81.49	74.76	77.98
train=economy, test=novel	77.67	75.04	76.33

Table 1: Results for different feature sets (top) and textual domains (bottom).

be seen when comparing our results with the performance of an unsupervised tree aligner. Running the subtree aligner described in [17] on the same data set yields an F-score of 57.57% which is similar to the scores we obtained when using the same features only. When using a better model for estimating lexical probabilities (more data: Europarl+SMULTRON) the performance improves only slightly to about 58.64%. We, can see that the additional features and the optimization of their contributions through machine learning has a strong positive effect on the performance. A drawback of supervised techniques is that we have to drop the generality of the unsupervised approach and require aligned training data to build language pair specific models. Furthermore, there is a risk of overfitting. In order to test the flexibility of our approach we ran several cross-domain experiments using the two domains present in the Smultron treebank. The results are presented in the lower part of table 1. As we can see there is only a slight drop in performance when training on a different textual domain. However, we still have reasonably high accuracy which is certainly encouraging especially considering the effort of human annotation necessary when preparing appropriate training data.

Finally, we also looked at the training curves with varying amounts of training data. For this we used about a third of the corpus (2667 links) for testing and trained on parts of the remaining data. Figure 1 shows the impact of training size on F-scores for three feature settings.

We can clearly see that the aligner yields a high performance already with little amounts of training data. Depending on the features the training curve levels out already at training sizes way below 100 sentence pairs. This is especially apparent for the settings that include word alignment features. Their values bear a lot of positive link evidence on their own and corresponding weights do not need to be adjusted very much. Other features such as the binary label-pairs require larger amounts of training examples (which is not very surprising). However, the alignment performance does not seem to improve significantly after about 128 sentence pairs even with those features included (see, for example, “no wordalign features” in figure 1).

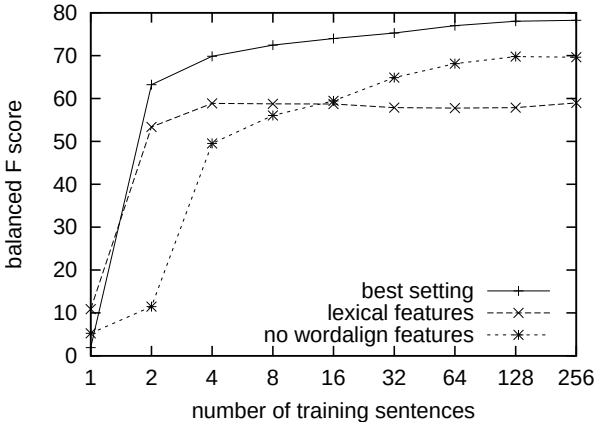

Figure 1: Training curve for various amounts of training data

4 Scaling Up

4.1 Creating a New Alignment Model

After tuning and testing our tree aligner as described above the next step is to apply the optimized model to large parallel treebanks. Unfortunately, to our knowledge no tree-aligned training corpus is available for the language pairs we are interested in (English-Dutch and French-Dutch). Therefore, the first step is to create a training corpus for our purposes. Fortunately, from our experiments we can conclude that only a small amount of aligned sentences is necessary to obtain reasonable results. Therefore, we decided to create a small aligned treebank to start with. In particular, we aligned 100 sentences from the English-Dutch parallel Europarl corpus [8] using the Stockholm Tree Aligner. We decided to follow the guidelines as proposed by [14] in the context of the Smultron parallel treebank project.

In order to speed up the process we decided to leave out all links between terminal nodes. This reduces the annotation effort dramatically not only in terms of number of links that have to be drawn but also in terms of alignment decisions. There are usually a lot of difficult decisions when aligning words with each other as we know from the literature on statistical MT and alignment competitions. A drawback is, of course, that we will miss this information when learning our alignment model. However, we believe that we will not lose a lot in performance if we take links between terminal nodes directly from existing word alignments.

For aligning the non-terminal nodes we decided to follow some guidelines as proposed by [14] in the context of the Smultron parallel treebank project:

- Align as many nodes as possible.
- They must be translatable to each other.
- Grammatical forms do not need to fit.

- Approximate translations are accorded fuzzy links. For example, one side may contain more general or more specific information.
- Pronouns are not linked to full noun phrases.
- Nodes with extra information that is missing and required on the other side are not linked. This ensures that a sentence is not linked to a verb phrase, for example.
- We only allow 1:1 links between non-terminal nodes.

We modified our implementation such that it can be run to consider only non-terminal nodes in training and alignment which is very straightforward. The software is also able to recognize existing links when aligning trees which is important when adding non-terminal links to a word-aligned parallel treebank. This is especially important for proper handling of history features and checking well-formedness criteria. Using these settings we trained a new model on our tiny set of training examples. Note that we used alignment features (for example, lexical probabilities) from a GIZA++ model trained on the entire Europarl corpus in order to get reliable estimates.

We also performed a sanity check with our newly created training corpus in order to see if the expected performance can be reached. For this we used 80 aligned sentences for training and the remaining ones for testing. This gives us a good idea about the expected outcome although the test set is way too small to give reliable evaluation scores. The results are as follows:

Precision	73.20%
Recall	86.89%
F	79.46%

As we can see, the scores are quite high especially in recall. This is partially due to the nature of our training material in which we prefer accurate links and tried to avoid fuzzy links as much as possible (in order to reduce the number of unreasonable/questionable links). Note that we only consider non-terminal nodes now in this evaluation. From this little experiment we may conclude that we can expect reasonable alignments even for our new language pair and a different textual domain.

4.2 Aligning Europarl

We are now able to align the entire Dutch-English part of the parallel Europarl corpus with the alignment model created as described above. For this we parsed both sides of the corpus with existing tools (Alpino for Dutch and the Stanford Parser for English) and converted the output of these parser to Tiger XML, which is used in our tree aligner implementation. The advantage of Tiger XML is that it supports even non-projective structures and it is also used by the Stockholm Tree Aligner which enables us to visualize alignment results easily (see figure 2).

Moreover we can even manually correct automatic alignments in this way which might be a valuable feature in future research.

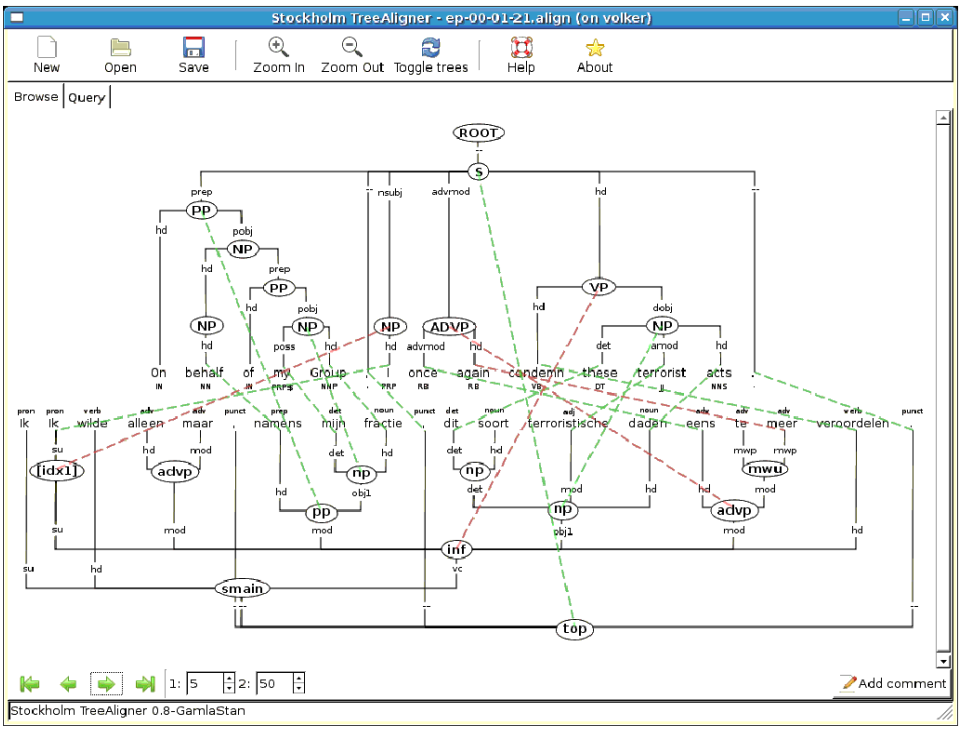

Figure 2: Automatic tree alignments visualized by the Stockholm Tree Aligner

After these pre-processing steps we proceeded in the following way: First we added all word alignments from the intersection of the two Viterbi alignments created by GIZA++. They were classified as "good" links assuming that they are very reliable. Next, we added links from the word alignment using the "grow-diag" symmetrization heuristics [9]. These links are less reliable and therefore were tagged as "fuzzy". Finally we ran the tree aligner to add non-terminal nodes to the parallel treebank.

There are some consequences of this alignment pipe-line architecture especially in combination with the well-formedness criteria. Word alignments often cause violations of these constraints and block the possibility of adding links on higher nodes. One solution to this problem is to disable the well-formedness check between terminal and non-terminal nodes and to allow violations at this level. However, this is not really desirable. Therefore, we use another mode in which existing word links compete with new incoming links in the same way as usual. For this we need to specify link scores for existing word alignments. In our experiments we simply defined arbitrary fixed scores (0.8 for links coming from the intersection and 0.4 for links coming from the grow-diag heuristics). As a con-

sequence, certain word alignments may disappear from the final result if they are overruled by other stronger links that do not meet the constraints with these particular word alignments (which is probably a good thing).

Using this setup we aligned the entire Dutch-English Europarl corpus. Note that we only apply the aligner to one-to-one sentence alignments and that some sentences were dropped due to parsing time-outs. However, we obtained a substantial amount of machine-aligned parse trees (see table 2).

	good ($p > 0.5$)	fuzzy	all
Europarl English-Dutch	20,109,174	5,205,048	25,314,222
non-terminals	6,999,851	3,821,970	10,821,821
terminals	13,109,323	1,383,078	14,492,401
links/sentence	19.01	4.92	23.93

Table 2: Links in 1,057,875 aligned parse trees from Europarl Dutch-English

5 Conclusions

We have presented a discriminative tree aligner that can be trained on small amounts of hand-aligned training data using a rich feature set. With this tool we achieve state-of-the-art performance in tree-to-tree alignment that can be used to align parallel treebanks on a larger scale. We are currently aligning large automatically parsed parallel corpora which we will use as a resource in the development of a syntax-oriented data-driven machine translation system.

References

[1] Lars Ahrenberg. LinES: An English-Swedish parallel treebank. In *Proceedings of the 16th Nordic Conference of Computational Linguistics*, 2007.

[2] Hal Daumé III. Notes on CG and LM-BFGS optimization of logistic regression. Implementation available at `http://hal3.name/megam/`, 2004.

[3] T.G. Dietterich. Machine learning for sequential data: A review. In T. Caelli, editor, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 2396 of *Lecture Notes in Computer Science*, pages 15–30. Springer, 2002.

[4] Daniel Gildea. Loosely tree-based alignment for machine translation. In *Proceedings of ACL-03*, pages 80–87, Sapporo, Japan, 2003.

[5] Annette Rios Gonzales, Anne Göhring, and Martin Volk. A Quechua-Spanish parallel treebank. In *Proceedings of TLT7*, 2009.

[6] Declan Groves, Mary Hearne, and Andy Way. Robust sub-sentential alignment of phrase-structure trees. In *Proceedings of COLING 2004*, pages 1072–1078, Geneva, Switzerland, 2004.

[7] Sofia Gustafson-Čapková, Yvonne Samuelsson, and Martin Volk. SMULTRON (version 1.0) - The Stockholm MULtilingual parallel TReebank. http://www.ling.su.se/dali/research/smultron/index.htm, 2007. An English-German-Swedish parallel Treebank with sub-sentential alignments.

[8] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*, 2005.

[9] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL-07*, Prague, Czech Republic, 2007.

[10] Alon Lavie, Alok Parlikar, and Vamshi Ambati. Syntax-driven learning of sub-sentential translation equivalents and translation rules from parsed parallel corpora. In *Proceedings of the ACL-08: HLT Second Workshop on Syntax and Structure in Statistical Translation*, pages 87–95, Columbus, Ohio, 2008.

[11] Joakim Lundborg, Torsten Marek, Maël Mettler, and Martin Volk. Using the Stockholm TreeAligner. In *Proceedings of TLT6*, pages 73–78, Bergen, Norway, 2007.

[12] I. Dan Melamed. *Empirical Methods for Exploiting Parallel Texts*. The MIT Press, 2001.

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

[14] Yvonne Samuelsson and Martin Volk. Alignment tools for parallel treebanks. In *Proceedings of GLDV Frühjahrstagung 2007*, 2007.

[15] Yvonne Samuelsson and Martin Volk. Automatic phrase alignment: Using statistical n-gram alignment for syntactic phrase alignment. In *Proceedings of TLT6*, pages 139–150, Bergen, Norway, 2007.

[16] Wei Wang, Jin-Xia Huang, Ming Zhou, and Chang-Ning Huang. Structure alignment using bilingual chunking. In *Proceedings of COLING 2002*, pages 1–7, Taipei, Taiwan, 2002.

[17] Ventsislav Zhechev and Andy Way. Automatic generation of parallel treebanks. In *Proceedings of COLING 2008*, pages 1105–1112, 2008.