Bottom-up transfer in Example-based Machine Translation

Vincent Vandeghinste
Centrum voor Computerlinguïstiek
Katholieke Universiteit Leuven
Belgium
`vincent@ccl.kuleuven.be`

Scott Martens
Centrum voor Computerlinguïstiek
Katholieke Universiteit Leuven
Belgium
`scott.martens@ccl.kuleuven.be`

Abstract

This paper describes the transfer component of a syntax-based Example-based Machine Translation system. The source sentence parse tree is matched in a bottom-up fashion with the source language side of a parallel example treebank, which results in a target forest which is sent to the target language generation component. The results on a 500 sentences test set are compared with a top-down approach to transfer of the same system, with the bottom-up approach yielding much better results.

1 Introduction

In machine translation, the use of linguistics had all but disappeared during the rise of the *statistical machine translation* (SMT) paradigm, but as "pure" SMT is reaching its ceiling, the pendulum is swinging back towards the use of linguistics in MT, even within the SMT paradigm, as demonstrated by the Workshops on Syntax and Structure in SMT (among others Wu and Chiang, 2009).

The MT engine which is used in this paper is a syntax-based *example-based machine translation* (EBMT) system.

It is *example-based* as it uses a large set of translation examples (a parallel corpus) as training data to base its decisions on and it is *syntax-based* as the data in the parallel corpus is annotated with syntactic parse trees, both on the source and the target side. Input sentences are syntactically analysed, and the system generates target language parse trees where all ordering information is removed. These serve as input for the target language gen-

eration component, which determines the output sentences.

The system can also be considered a *rule-based transfer* system, as it conforms to the general architecture of a rule-based system, using source language syntactic analysis, syntactic transfer rules and a dictionary (lexical transfer rules), and a target language generation component. Both syntactic and lexical transfer rules are automatically induced from a parallel corpus.

2 Related Work

We compare the transfer component described in this paper with the transfer component of Vandeghinste and Martens (2009).

The general approach towards MT is quite similar to Data-Oriented translation (DOT) (Poutsma, 1998; Hearne, 2005), differing in the fact that we use rule-based or probabilistic context-free parsers whereas they use Data-Oriented Parsing (Bod, 1992), and the DOT approach was only tested on small corpora and a limited domain, whereas we intend a general news domain using large corpora.

There are also similarities with the work of Ambati et al. (2009). They use *synchronous context-free grammars (SCFGs)* (Aho and Ullman, 1969), which limit the depth of the transfer rules to 2, whereas the approach described in this paper does not set a limit to the maximum depth of a transfer rule, just like the *synchronous tree-substitution grammars (STSGs)*, as described in Zhang et al. (2007). The difference between our system and STSGs is the fact that we build a target language tree without using any ordering information, since this is handled in the decoding step: the target language generation component (Vandeghinste, 2009).

Our general approach is also similar to the

example-based MT-engine described by Kurohashi (2009), differing in the fact that Kurohashi uses dependency trees and we combine information from phrase structure trees and dependency trees.

3 System Description

The example-based machine translation system has an architecture very similar to that of rule-based transfer systems. An input sentence is analyzed by a source language parser. The source language parse tree is converted by the transfer component into a target language forest that represents all possible target language parses that are considered translation candidates. The target language generation component turns this forest into a ranked set of sentences, each with their weight.

3.1 Syntactic Analysis

The system reuses existing parsers for both source and target language analysis. As we are translating from Dutch to English, the system uses the Alpino parser (Van Noord, 2006) for Dutch, which outputs results in an xml-format combining phrase structure information with dependency information; and the system uses the Stanford parser (Klein and Manning, 2003) for English, which gives a phrase structure tree and an additional dependency tree (de Marneffe et al., 2006). Both parsers are freely available.

3.2 Preprocessing the parallel corpus

The system was trained on the sentence-aligned Europarl corpus version 3 (Koehn, 2005).

The source language parser is used to parse the source side of the parallel corpus in preprocessing, as well as the input sentence during actual translation. The target language parser is only used to preprocess the target side of the parallel corpus. This results in a parallel treebank, on which more details can be found in Tiedemann and Kotzé (2009a). This treebank is word aligned with GIZA++ (Och and Ney, 2003) and node aligned using a discriminative approach to tree alignment (Tiedemann and Kotzé, 2009b).

3.3 Bottom-up transfer

The transfer component takes the source language parse tree and matches the nodes in that tree with nodes on the source language side of the parallel treebank. The corresponding target side fragments are recombined into target language trees. All possible output trees of this component are merged into a target language forest.

Vandeghinste and Martens (2009) describe a *top-down* transfer component which leads to unsatisfactory results. We have investigated a *bottom-up* transfer component instead. As this paper is mainly about this transfer component, we describe it more thoroughly in section 4, and compare the scores of this bottom-up approach with the scores of Vandeghinste and Martens (2009) on the top-down approach, using the same test set.

3.4 Target language generation

Target language generation (TLG) is the component that converts the target language forest into a set of target language sentences, ordered by their confidence weight. The system uses the same target language generation component as Vandeghinste and Martens (2009). It is an improved version of the TLG module of Vandeghinste (2009).

The target language forest, which is the output of the transfer component does not contain any target language ordering information. For each parent node, the actual order of the daughters is determined by the target language model. It determines word and constituent ordering and plays an important role in lexical selection.

The target language model is trained on the English part of Europarl. From the parse trees in the treebank, a set of context-free rules is extracted, using the phrase category labels on the left-hand side. The TLG model is not restricted to parts-of-speech, the phrase category labels or the tokens on the right-hand side as different abstraction levels are distinguished. For English, these are: dependency relations (Rels), syntactic categories for non-terminal nodes and the parts-of-speech for terminal nodes (Cat/Pos), the dependency relations together with the syntactic categories/parts of speech (Cat+Rel), the dependency relations together with the syntactic categories/parts of speech as well as the head token information (Cat+Rel+Token).

Traversing the target language trees in the forest depth-first the TLG module checks whether it finds rewrite rules at the least abstract level (Cat+Rel+Token). If this is not the case, it checks at the next level and so on until a solution is found allowing to estimate the probability of different orderings of the daughter of the node it is looking at.

Depending on the beam size and a number of other cutoff parameters, it selects the *n* most probable by looking at the relative frequency of occurrence of the different patterns in the training data.

The parameters that allow us to investigate the trade-off between quality and time of processing are the following:

- *Beam size*, a.k.a. histogram pruning;

- *Cutoff factor*, a.k.a. threshold pruning;

- *Maximum Combinations* sets a limit to the number of combinations investigated, ordered by weight. Imagine a node with three daughters, and for each daughter an average of ten solutions where found, then this combines into 1000 combinations.

- *Maximum Permutations* sets a limit to the number of permutations under investigation, when no solutions are found in the database. All permutations of that node are generated. This can lead to very high numbers, as the number of permutations is the faculty of the number of daughters of a node

For each of these parameters, before any cutoff happens all alternatives are ordered according to their weights.

4 Bottom-up Transfer

In response to the shortcomings of the top-down model of Vandeghinste and Martens (2009), we proposed and implemented an alternative transfer strategy, one that proceeds from the bottom upwards, starting with translations of words and phrases and then selecting among the translations further up in the parse tree on the basis of the translations discovered at the bottom. The logic of this approach is that it would be better to confidently translate words and phrases in source sentences, and then use those translations to constrain the choice of structures above. In this way, errors might propagate upwards but not downwards, where they had proven to force the transfer engine to make unlikely and unacceptable translations.

4.1 Indexed treebanks and virtual rules

To do this, we did not extract rules of predetermined depths of trees like Vandeghinste and Martens (2009) but embarked on constructing a system of virtual rules, in which the treebank itself

would be consulted, on the fly, to identify translations at all levels.

For lexicalized translations, where an entire phrase that appears as a constituent in the parse tree also appears in the treebank, we deployed the solution originally proposed by Luccio et al. (2004). Trees are reordered so that the children of each node in the parse tree appear in a fixed lexicographic order, ignoring the original word order. These trees are then rewritten as strings, using what Luccio et al. (among others) refer to as a depth-first order. If the tree under some node in the parse tree is identical to the tree under some node in the treebank, and if both are converted to depth-first format, then there is a substring in the treebank that is identical to the one representing that portion of the parse tree. This is called bottom-up subtree matching. Given two trees, a bottom-up subtree (Figure 1) match is one that, if it matches any node, also matches all the descendants of that node.
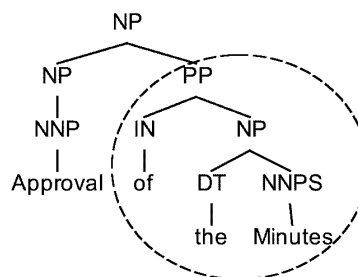

Figure 1: "of the Minutes" is an example of a bottom-up subtree

Performing bottom-up subtree matching is similar to the ideas behind subsentential translation memories: each match is to a linguistically motivated phrase within sentences, and where a match is found and that match aligns to some subtree in the target language, translation can proceed by copying that target language subtree.

Finding string matches quickly in large texts has a well-known solution: the suffix array, which identifies matches in indexed strings in sublinear time (Manber and Myers, 1990). By converting the problem of subtree discovery into a string matching problem, we can extract transfer rules from the treebank for any node very quickly.

For transfer of the upper portions of the parse tree, we found that we could generalize the rule construction method described for top-down trans-

```
          NP                    ........ DT $ the # $
                               .:IN $ of # $
     NP       PP              .::NNPS $ Minutes # $
                            .::·NNP $ Approval # $
    NNP    IN    NP       ...::··NP $ DT NNPS # $ the # Minute # $
                         .:::···
  Approval  of   DT  NNPS :::···NP $ NNP # $ Approval # $
                        .::···:NP $ NP PP # $ NNP # IN NP # $ Approval # of # DT NNPS # $ # # the # Minutes # $
             the  Minutes :::···PP $ IN NP # $ of # DT NNPS # $ # the # Minutes # $
```
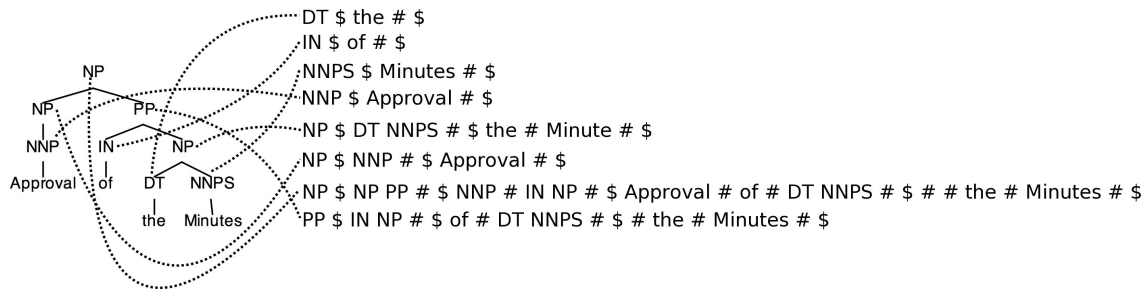
Figure 2: Constructing and sorting breadth-first representations of the subtrees of the example parse from Figure 1.

fer as described by Vandeghinste and Martens (2009) by modifying the string matching techniques used for bottom-up matching, and then dispensing with rule-sets and using the treebank to perform those transfers as well. Instead of converting trees into strings using depth-first representations, we took each non-leaf node in the source language treebank and converted it into a string using a breadth-first method inspired by Chi et al. (2005).

Converting a tree to a breadth-first string representation (BFSR) requires two extra symbols - represented here by "#" and "$" - one to indicate the exhaustion of the children of some node, and the second to indicate the exhaustion of the nodes at a particular depth in the tree. The process proceeds by reading breadth-first through the tree starting at the root, appending node labels to an initially empty string. When all the children of a node have been exhausted, the "#" symbol is added, and when all the nodes at some depth have been exhausted, the "$" is added. This maps each source language node in the treebank to a string, as shown in Figure 2. These string representations can be trivially converted back into trees and stand in a one-to-one correspondence with the trees that generate them. Note that BFSRs are sortable and that if any two subtrees are identical from the root down to some particular depth, then the BFSRs of those two subtrees share a common prefix. By organizing them into a sorted array, we can quickly match any subtree in a new parse tree to all subtrees in the treebank that share the same upper part. This makes search using string indexing methods feasible.

In this implementation, a BFSR was constructed for each non-terminal node in the treebank, consuming space proportionate to the mean square of the size of each sentence. Then these string representations were sorted using *quicksort* (Hoare

1962). Alternative and possibly more efficient strategies for indexing these representations are also feasible, based on the expansive literature on suffix tree and suffix array construction. These would be equivalent in terms of results, and come with various tradeoffs in preprocessing time and space.

4.2 Matching the source language tree with the examples

The system proceeds by, first, checking for bottom-up matches in the source language tree indices. Finding one is equivalent to finding a subsentential match in a translation memory system. Figure 3 shows a possible set of bottom-up matches.
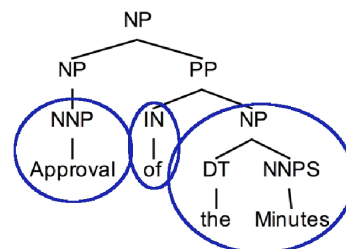

Figure 3: Bottom-up matching finds all phrases and words that have matches in the treebank

The transfer engine then tries to identify top-down matches for the remaining upper portion of the tree, and rejects all matches that are incompatible with the bottom-up matches already found, as in Figure 4. Top-down matching proceeds by constructing a BFSR for each unmatched node in the source parse tree, as described in section 4. For each such BFSR, the transfer engine searches the sorted index of BFSRs from the treebank for the
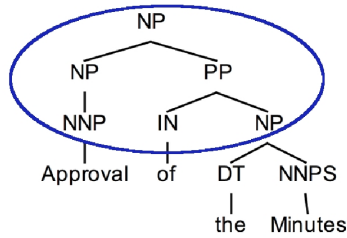
Figure 4: Top-down matching looks for structures in the source language treebank matching the remaining part of the translation. Note that the leaves of the subtree being matched using top-down methods must all be at the same depth.
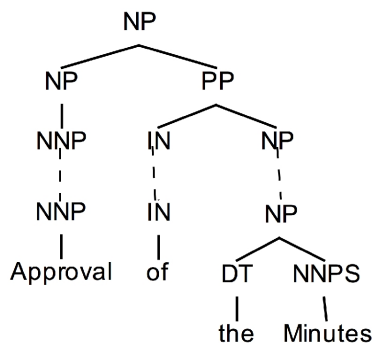

Figure 5: Each top-down match is finally connected to the bottom-up matches

entries that share the longest common prefix with it.

The treebank alignment information discussed in 3.2 is used to align the source language nodes pointed to by the sorted index with their corresponding target language nodes. Those target language parses are then directly searched for subtrees that can join together the bottom-up matches already found. When there are too many matching nodes, a random sample is searched. The resulting target language subtrees are then combined with the bottom-up matches already found to form a target language tree, as in Figure 5.

This procedure is performed recursively over the parse tree, until the entire tree is translated.

Where a word is missing from the treebank, or has no target language alignment, the fall-back translation strategy is the same as for the top-down approach from Vandeghinste and Martens (2009): The part-of-speech or other information is translated and the word copied over directly. However, the search for structural translations of the upper

parts of parse trees may also fail to find a match. In those cases, two strategies are considered.

First, a special target language index is constructed that contains the labels - phrase categories or parts-of-speech - for each non-leaf node in the target language treebank. When no top-down match can be found, this database is searched for any target language node whose children are identical to the labels of bottom-up matches whose roots are siblings in the source language parse and whose own label is a likely match for the source language phrase label that appears above them. The transfer engine then uses that shallow, two level tree to translate the corresponding source subtree.

For example, if there was no transfer found for the upper portion of the tree in Figure 4, the transfer engine would look for nodes in the target language index that have an IN and an NP as children, and that are likely to correspond to the Dutch phrase category label pp.

This fall-back strategy tends to produce trees that closely hew to the structure of the source.

When even this strategy fails, the transfer engine assumes that nodes that are siblings in the source have translations that are siblings in the target language. So, when no other transfer rule is available, it selects the target language node label that most corresponds to the source language parent, and then guesses which of the target language child nodes is likely to be the head of that phrase, based on what labels are usually heads for that type of phrase.

Translating from the bottom-up in this manner is closely related to classical parsing strategies which build tree structures up from the bottom.

5 Evaluation

We evaluated our system, using well-known automated MT metrics, like BLEU (Papineni et al. 2002), NIST (Doddington 2002), and TER (Snover et al. 2006), as well as WER (word error rate), PER (position independent word error rate), and CER (character error rate). We have used the same evaluation test set as Vandeghinste and Martens (2009) consisting of 500 Dutch sentences, with two reference translations for each sentence. To give an idea about the difficulty of the test set, it scored 29.96 BLEU on Moses (Koehn et al. 2007) trained on the same sentences of Europarl as used in our system and 38.82 BLEU on Google trans-

late.

We evaluated the bottom-up system in three conditions:

1. Smallbeam: In target language generation, we use a beam size of 10, a cutoff factor of 50, a maxcomb of 100 and a maxperm of 100

2. Largebeam: In target language generation, we use a beam size of 100, cutoff factor of 500, maxcomb of 200 and a maxperm of 200

3. Dummy: Only bottom-up transfer of matching words, as described in section 4.2. Source word order is retained and the target language generation module favours orders that are close to the source order, when all else is equal.

The results are shown in Table 1. The results of Vandeghinste and Martens (2009) are added in the *Top-down* row.

We also compared with Moses (Koehn et al., 2007) trained on the same data, and using the same word-alignments. Due to the source language parser of our system which puts all punctuation outside the actual parse tree, our system does not handle punctuation (yet). To get a better comparison with the state-of-the art of Moses, in table 1 we remove all punctuation from its output as well.

The results for the bottom-up approach to transfer are a lot better than the results for the top-down system. There is a relative rise of 52.7% in BLEU score when comparing the best conditions of the top-down and the bottom-up approach.

Furthermore, we can see that the difference with Moses (without punctuation) has become very small when considering the PER metric, which indicates the position-independent word-error rate. This is important as it indicates the fact that concerning lexical selection our early prototype system scores only marginally worse than the state-of-the-art.

Comparing the scores with the *Dummy* condition gives an indication of the influence of structural transfer in both lexical selection as well as re-ordering of the output. All scores consistently indicate that structural transfer contributes substantially to better lexical selection. When comparing the PER score of the *Dummy* condition with the PER score of the *Top-down* approach, it is clear

that lexical selection is better bottom-up, even when the influence of structural transfer has been removed as is the case in the *Dummy* condition.

Concerning the beam size in target language generation we can say that there is no significant difference in results of the two conditions, but it is significantly faster to process the sentence in target language generation for the *Smallbeam* condition.

6 Conclusions and Future Work

An important conclusion from the results is the fact that in lexical selection, our results are similar to those of Moses. There are still a few differences, for instance in the treatment of separable verbs, and we have implemented solutions for this which are not yet reflected in these results. This will require a complete reprocessing and realigning of the parallel treebank, which is a very time consuming and computationally heavy process.

The influence of the structural transfer is large and positive, and therefore indicates that we should work on that aspect of our engine more: we can test different parameter settings, and in future versions of the system, we also want to include partial subtree matching, which should greatly improve the coverage of the parallel corpus with respect to structural transfer.

Improvements to the virtual transfer rule system are a major research direction for this project. The current scheme, which searches the aligned treebank directly, using sampling in many cases, is in the worst case linear in performance time on the size of the treebank or the sample size where sampling is used. Using subtree indexing (Chi et al., 2005; Martens, 2009), we hope to reduce this time dramatically.

The virtual rule system implemented here constitutes a *regular tree grammar (RTG)*, which is weakly equivalent in generating capacity to a context-free grammar (CFG) (Thatcher, 1967; Rounds, 1970), that is to say that the trees generated by every RTG yield a set of strings for which some CFG exists that generates them. Its principal benefit is that, by generating trees, it separates the generation of target language strings from the induction of target language linguistic structures. However, the limitations of CFGs and comparable tree grammars are well-known. Context-free tree grammars are weakly equivalent to indexed grammars (Rounds, 1970), which provide a much larger set of options, at the cost of NP-complete process-

Condition	BLEU	NIST	WER	CER	PER	TER
Top-down	13.53	5.70	76.20	61.91	52.39	70.36
Dummy	12.49	6.01	78.75	63.83	50.05	70.69
Smallbeam	**20.65**	**6.44**	70.34	55.37	**48.96**	63.72
Largebeam	20.59	6.43	**70.10**	**55.12**	48.98	**63.54**
Moses No Punct.	26.72	6.94	60.53	45.65	47.82	58.07

Table 1: Evaluation Results

ing times, just like the indexed grammars. The tree-adjoining grammar (TAG) formalism (Joshi et al., 1975) limits context-sensitive generation to the monadic context-free tree grammars (Mönnich, 1997; Fujiyoshi,2004), and other subsets of tree grammars are available for linguistic formalisms (Knight and Graehl, 2005).

Extending the machinery for syntactic transfer beyond RTGs to more powerful formalisms is a major future research area for this project. Notably, work is in progress to extend the virtual transfer rule system to support non-deleting tree rules (Knight and Graehl, 2005) which can be efficiently extracted from aligned treebanks using data mining techniques (Martens, 2009).

There is also room for improvement in subsentential alignments. We will investigate whether there are other alignments possible which will lead to better results. For now we are using a first version of the alignments, but the work we have done up to now has given us a great deal of information about how we might improve the alignment. This is not reflected in the alignments as they are now, as this requires to reapply the time consuming alignment processing of the parallel data.

The evaluation of this system has shown some encouraging results, and detailed error analysis has shown some of the paths to follow in the future.

We will first of all try our approach on other language pairs and see whether the conclusions still hold.

Apart from that, we are working on an indexing system which will allow us to work with partial subsentential matches instead of full subtree matching, which will have a large effect on the coverage of the parallel treebank, as well as on the speed of the transfer engine, which is rather slow as it is now. This will also solve a number of translation issues for which the current system cannot generate a correct translation unless the whole phrase is found in the parallel treebank.

We will also investigate the effect of enlarg-

ing the treebanks used, both parallel and monolingual, including the translation memories we have received from a translation company.

In general, we can conclude that we have come to a point where we are reasonably satisfied with the transfer engine, which can serve in the first version of the MT system, but there is plenty that remains to be done to further improve the system.

References

Abouelhoda, M., Kurtz, S., and Ohlebusch, E. (2004). Replacing Suffix Trees with Enhanced Suffix Arrays. *Journal of Discrete Algorithms*, 2(1):53-86.

Aho, A., and Ullman, J. (1969). Syntax directed translations and the pushdown assembler. *Journal of Computer and System Sciences*, volume 3(1):37-56.

Ambati, V., Lavie, A., and Carbonell, J. (2009). Extraction of Syntactic Translation Models from Parallel Data using Syntax from Source and Target Languages. In: *MT Summit XII Proceedings of the twelfth Machine Translation Summit*. Ottawa, Ontario, Canada. pp. 190-197.

Bod, R. (1992). A Computational Model of Language Performance: Data-Oriented Parsing. In *Proceedings of the fifteenth International Conference on Computational Linguistics (COLING'92)*. International Committee on Computational Linguistics. Nantes, France. pp. 855-859.

Chi, Y., Nijssen, S., Muntz, R., and Kok, J. (2005). Frequent Subtree Mining An Overview. *Fundamental Informatics, Special Issue on Graph and Tree Mining*. pp. 1001-1038.

de Marneffe, M., MacCartney, B., and Manning, C. (2006). Generating Typed Dependency Parses from Phrase Structure Parses. In: *Proceedings of the 5th edition of the International Conference on Language Resources and Evaluation (LREC)*. Genoa, Italy.

Doddington, G. (2002). Automatic Evaluation of Machine Translation Quality using N-gram Co-occurrence

Statistics. In: *Proceedings of the Second Human Language Technology Conference (HLT)*. Morgan Kaufmann. San Diego, USA. pp. 138-145.

Fujiyoshi A. (2004). Epsilon-free grammars and lexicalized grammars that generate the class of the mildly context-sensitive languages. In: *Proceedings of the 7th International Workshop on Tree Adjoining Grammar and Related Formalisms*. Vancouver, pp. 16-23.

Hearne, M. (2005). *Data-Oriented Models of Parsing and Translation*. PhD thesis. Dublin City University. Dublin, Ireland.

Hoare, C. (1962) Quicksort, *Computer Journal* 5, pp. 10-15.

Joshi, A., Levy, L., and Takahashi, M. (1975) Tree adjunct grammars, *Journal of Computer and System Sciences* 10, pp. 136-163.

Kärkkäinen J., and Sanders P. (2003) Simple linear work suffix array construction, In: *Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP'03)*. Eindhoven, Netherlands. pp. 943-955.

Klein, D., and Manning, C. (2003). Accurate Unlexicalized Parsing. In: *Proceedings of 41st Annual Meeting of the Association of Computational Linguistics (ACL)*. Sapporo, Japan. pp. 423-430.

Knight, K. and Graehl, J. (2005). An Overview of Probabilistic Tree Transducers for Natural Language Processing. In: *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics*

Koehn, P. (2005). Europarl. A parallel corpus for statistical machine translation. In: *MT Summit X: Proceedings of the tenth Machine Translation Summit X*. Phuket, Thailand. pp. 79-97.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer D., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In: *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*. Prague, Czech Republic. pp. 177-180.

Kurohashi, S. (2009). Fully syntactic example-based machine translation (abstract). In *Proceedings of the 3rd International Workshop on Example-based Machine Translation (EAMT)*. Dublin City University, Dublin, Ireland. p. 1.

Luccio, F., Enriquez, A., Rieumont, P., and Pagl, L. (2004). *Bottom-up subtree isomorphism for unordered labeled trees*. Technical Report TR-04-13, Universit Di Pisa. Pisa, Italy.

Manber, U., and Myers, G. (1990). Suffix arrays: a new method for on-line string searches. In: *SODA 90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia. pp.319-327.

Martens, S. (2009). Quantitative Analysis of Treebanks using frequent subtree mining methods. In: *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing (TextGraphs-4)*. pp. 84-92.

Mönnich, U. (1997). Adjunction as substitution. In: G.-J. Kruiff, G. Morrill, and D. Oehrle (eds.) *Formal Grammar*. pp. 169-178.

Och, F., and Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics* 29 (1), pp. 19-51.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2002). BLEU: a method for automatic evaluation of Machine Translation. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. Philadelphia, USA. pp. 311-318.

Poutsma, A. (1998). Data-Oriented Translation. In: Ninth Conference of Computational Linguistics in the Netherlands (CLIN). Leuven, Belgium.

Rounds, W. (1970). Tree oriented proofs of some theorems in context-free and indexed languages. In: *Proceedings of the 2nd ACM Symposium on Theory on Computing*, 109-116.

Snover, M., Dorr, B., Schwartz, R., Micciula, L, and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In: *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA)*. Cambridge, USA. pp. 223-231.

Thatcher, J. (1967). Characterizing derivation trees of context-free grammars through a generalization of finite automata theory" *Journal of Computer and System Sciences*, volume 1, 317-322.

Tiedemann, J., and Kotzé, G. (2009a). Building a Large Machine-Aligned Parallel Treebank. In: *Proceedings of the 8th International Workshop on Treebanks and Linguistic Theories (TLT)*. Milan, Italy. pp. 197-208.

Tiedemann, J., and Kotzé, G. (2009b). A Discriminative Approach to Tree Alignment. In: *Proceedings of Recent Advances in Natural Language Processing*. Borovets, Bulgaria. pp. 33-39.

Vandeghinste, V., and Martens, S. (2009). Top-down Transfer in Example-based MT. In *Proceedings of the 3rd International Workshop on Example-based Machine Translation*. Dublin City University, Dublin, Ireland. pp. 69-76.

Vandeghinste, V. (2009). Tree-based Target Language Modeling. In: *Proceedings of the 13th Annual Conference of the European Association for Machine Translation (EAMT)*. Barcelona, Spain. pp. 152-159.

van Noord, G. (2006). At Last Parsing Is Now Operational. In: *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*, Leuven, Belgium. pp. 20-42.

Weiner, P. (1973). Linear pattern matching algorithm. In: *Proceedings of the 14th Annual IEEE Symposium on Switching and Automata Theory*, pp. 1-11.

Wu, D., and Chiang, D. (2009). *Proceedings of the 3rd Workshop on Syntax and Structure in Statistical Translation*.

Zhang, M., Jiang, H., Aw, A., Sun, J., Li, S., and Tan, C. (2007). A tree-to-tree alignment-based model for statistical machine translation. In: *Proceedsing of MT Summit XI*, pp. 535-542.