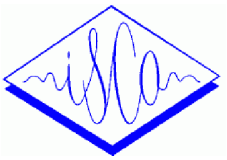
ISCA Archive
http://www.isca-speech.org/archive



International Workshop on
Spoken Language Translation 2011
San Francisco, CA, USA
December 8-9, 2011

Soft String-to-Dependency Hierarchical Machine Translation

Jan-Thorsten Peter, Matthias Huck, and Hermann Ney

Human Language Technology and Pattern Recognition Group
RWTH Aachen University, Aachen, Germany
{peter,huck,ney}@i6.informatik.rwth-aachen.de

Daniel Stein

Fraunhofer IAIS
St. Augustin, Germany
daniel.stein@iais.fraunhofer.de

Abstract

In this paper, we dissect the influence of several target-side dependency-based extensions to hierarchical machine translation, including a dependency language model (LM). We pursue a non-restrictive approach that does not prohibit the production of hypotheses with malformed dependency structures. Since many questions remained open from previous and related work, we offer in-depth analysis of the influence of the language model order, the impact of dependency-based restrictions on the search space, and the information to be gained from dependency tree building during decoding. The application of a non-restrictive approach together with an integrated dependency LM scoring is a novel contribution which yields significant improvements for two large-scale translation tasks for the language pairs Chinese–English and German–French.

1. Introduction

String-to-dependency hierarchical machine translation employs target-side dependency features to capture syntactically motivated relations between words even across longer distances. It is based on the hierarchical phrase-based paradigm [1] and implements enhancements that allow for an integration of knowledge obtained from dependency parses of the training material. Dependency trees over translation hypotheses are built on-the-fly during the decoding process from information gathered in the training phase and stored in the phrase table. A dependency language model can be applied to rate the quality of the constructed tree structures.

In initial publications on the topic [2, 3], a restriction of the phrase inventory to a subset of phrases which meet certain validity conditions concerning the dependency relations is proposed. Phrases with dependency structures that are not suitable for the construction of a well-formed dependency tree are excluded beforehand. Additional merging constraints apply during decoding. In later works [4, 5], heuristics are proposed that enable assembling of malformed dependency structures as well, thus permitting the utilization of the full phrase inventory of the standard hierarchical approach. Validity and tree well-formedness conditions are modeled in a soft way as features in the log-linear model. Here, the dependency language model is however included in an n-best reranking framework only.

This paper aims at filling the gap by investigating string-to-dependency hierarchical translation with and without restrictions, and by comparing dependency LM reranking methods with dependency LM scoring integrated into the decoder. In particular, we explore the following aspects:

- In an n-best reranking framework, only a limited amount of fully generated sentences is presented to the reranking models. We evaluate whether the dependency LM works better in decoding or in reranking.
- The constructed dependency tree is probably erroneous, but so is a parse obtained directly with a dependency parser on a grammatically malformed hypothesis. We analyze whether in a dependency LM reranking framework a direct parsing of the n-best hypotheses performs better than tree building during decoding.
- Restrictions on the phrase table entries as well as on the allowed combination of phrases during decoding might prevent possibly beneficial hypotheses. We investigate whether a soft, i.e. feature-based, approach yields improvements over a restrictive method that guarantees tree well-formedness. We analyze which limitations, if any, are more useful when compared to a non-restrictive approach.
- In the soft approach, the feature set of the log-linear model of the baseline hierarchical system is augmented with additional dependency-based features that can be categorized in two groups: those associated with the tree building process and those related to the dependency LM. We study how dependency tree building features and dependency LM each perform in isolation.
- Usually trigrams are used for the dependency language model. We analyze the typical dependency tree structures found in our data and, based on the findings, explore which dependency language model order is appropriate.

Results are presented in BLEU [6] and TER [7] on a NIST Chinese–English subset and on a German–French corpus.

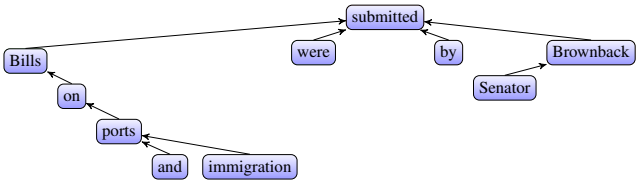


Figure 1: Dependency tree of an English sentence.

2. Related Work

Besides the authors mentioned in the previous section, several other groups have introduced dependency-based extensions to their machine translation frameworks in recent years.

Among them, Galley & Manning [8] perform dependency parsing during decoding by lowering the parsing time to quadratic complexity. The authors work on the NIST Chinese–English corpus and show significant improvements over the baseline. Bach et al. [9] rely on source-side dependency information for reordering information on the tree structure. Similar to lexicalized reordering models, the tree sub-structure is used to employ features in a phrase-based system, for English–Spanish and English–Iraqi. Gao et al. [10] propose soft reordering constraints based on the source-side dependency structure in a hierarchical setting with improvements on the NIST Chinese–English task. Xie et al. [11] also work with source-side dependencies and store the reordering information for each head-dependent. They employ a dependency-to-string translation model and compare its performance to a state-of-the-art hierarchical system, on the NIST Chinese–English task. Quirk & Menezes [12] present a treelet translation system with dependency projection from source to target and tree-based decoding.

3. Dependencies

A dependency models a linguistic relationship between two words, like e.g. the subject of a sentence that depends on the verb. Labelled by linguistic experts, large collections of training material have been used to derive parsers that are able to automatically label unknown sentences (e.g. [13]). While losing some accuracy, these parsers often have the option to allow for only one outgoing dependency relation per word, so that the dependency mapping within one sentence results in a dependency tree. See Figure 1 for an example.

3.1. Dependency Structures in Translation

String-to-dependency machine translation demands the creation of dependency structures over hypotheses produced by the decoder. Target-side dependency trees are also necessary for dependency LM scoring. There are various methods possible on how to obtain these trees. Parsing during decoding as well as parsing on a fully generated output hypothesis bear the risk that the accuracy of the tree is very low, since the output will likely be erroneous.

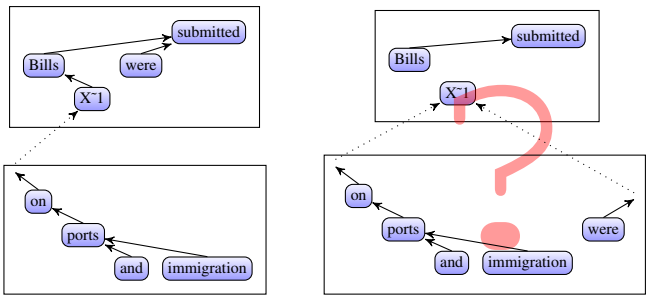


Figure 2: Example of a well-defined merge (left) and an ill-defined merge (right).

A different method is to parse the (hopefully grammatically sound) training material, and to carry the dependency structures over to the translated sentences by augmenting the entries in the phrase table with dependency information. However, the dependency structures seen on phrase level during phrase extraction are not guaranteed to be applicable for the assembling of a dependency tree during decoding. In Figure 2, we present an example with a well-defined merge of dependency structures during decoding, but also one where assembling a composed structure from the two parts causes problems. Many of the extracted phrases may be covered by structures where some of the dependencies contradict each other. A standard solution is to restrict the phrase table to only those entries that possess *valid* dependency structures, i.e. structures that comply with certain well-formedness requirements [2]. In an approach without hard restrictions, all kinds of structures are allowed, but invalid ones are penalized [4]. Merging heuristics allow for building of trees from malformed dependency structures as well.

3.2. Phrases With Valid Dependency Structures

A *fixed on head* dependency structure represents a structure where every word in the phrase depends only on other words in the same phrase, with the exception of one word which is defined as head. In a *floating with children* structure, the head is allowed to be outside of the phrase, but the dependencies inside the phrase cannot point to more than one head.

Formally, let $dep_d = r$ denote the relationship between a dependent word with index d on a regent word with index r. A phrase that ranges from word indexes i to j, which has the dependency structure $dep_{i\ldots j}$, is called *fixed on head* h, iff

$$\begin{aligned} \exists h \in [i,j],\ \text{s.t. } dep_h \notin [i,j] \\ \forall k \in [i,j]\ \text{s.t. } k \neq h, dep_k \in [i,j] \\ \forall k \notin [i,j], dep_k = h \vee dep_k \notin [i,j] \end{aligned} \tag{1}$$

and *floating with children* C for a non-empty set $C \subseteq \{i,\ldots,j\}$ iff

$$\begin{aligned} \exists h \notin [i,j],\ \text{s.t. } \forall k \in C, dep_k = h \\ \forall k \in [i,j]\ \text{s.t. } k \notin C, dep_k \in [i,j] \\ \forall k \notin [i,j], dep_k \notin [i,j] \end{aligned} \tag{2}$$

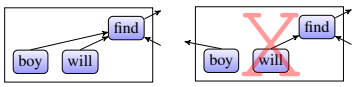


Figure 3: Fixed on head structure (left) and a counterexample (right).

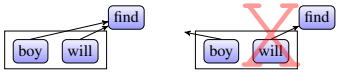


Figure 4: Floating with children structure (left) and a counterexample (right).

Note that we apply these dependency structures for hierarchical phrases as well, i.e. in the decoding step the indexes can represent terminal words as well as non-terminals. An example and a counterexample for each a *fixed on head* structure and a *floating with children* structure are shown in Figure 3 and Figure 4, respectively.

Based on the corpora used, only a small amount of phrases is marked as *fixed on head* structure, while the majority of valid phrases is marked as *floating with children* structures. We encounter large differences in the ratio of valid phrases between the Chinese–English task with 43.5% valid phrases and the German–French task with 29.3% valid phrases as shown in Table 1. This differs from the 19.2% for the Chinese–English task reported by Shen et al. [2]. One major influence to this discrepancy is probably that our table is filtered to contain only phrases from the translation sets which could cause a distortion of our reported ratios. Since this basically acts as a filter for very noisy and malformed training material, the dependency structures are in better shape. Other influences include the alignment and extraction heuristics applied to the training material.

In this work, we opt for keeping even phrases with invalid structures in our translation table since they still might contain valuable information. The number of invalid phrases used to create the translation is provided as additional feature function in order to equip the system with a means to control the tree assembling. We present an experimental comparison to the approach with hard restrictions in Section 5.1.3. A soft approach also means that we will not be able to construct a well-formed tree for all translations and that we have to cope with merging errors.

3.3. Dependency Tree Building During Decoding

During decoding, the previously extracted dependencies are used to build a dependency tree for each hypothesis. While in the optimal case the child phrase merges seamlessly into the parent phrase, often the dependencies will contradict each other and we have to devise strategies for these errors. An example of an ideal case is shown in Figure 5, and a phrase that breaks the previous dependency structure is shown in Figure 6. As a remedy, whenever the direction of a dependency within the child phrase points to the opposite direction of the parent phrase gap, we select the parental direction, but penalize the merging error via a count feature in the log-linear model. In a restrictive approach, the problem can be avoided by requiring the decoder to always obey the dependency directions of the extracted phrases while assembling the dependency tree.

Table 1: Amount of phrases with valid dependency structure as reported by Shen et al. and as observed in this work. Note that our phrase tables are filtered to contain phrases from the translation sets only.

	total	fixed on head	floating
Chinese–English [Shen et al.]	140 M	27 M	
Chinese–English	43 M	1.1 M	17.8 M
German–French	34 M	0.8 M	9.3 M

4. Dependency Language Model

Given a dependency tree of the target language, we are able to introduce language models that span over longer distances than shallow n-gram language models, via a language model on the dependency tree levels. More precisely, we compute several language model scores for a given tree: for each node as well as for the left and right-hand side dependencies of each node. The node model $Pr_T(w)$ is the probability for a word to be the head of a smaller dependency structure, modeled by a simple unigram language model. The left-hand side model $Pr_L(w_h|w_{h_{L1,L2,\ldots,L_n}})$ for a head node w_h with $w_{h_{L1,L2,\ldots,L_n}}$ dependent words that appear in previous sentence positions relative to the head, is computed with a regular n-gram language model. The right-hand side is modeled analogous for dependent words that appear in a later sentence position relative to the head. For each of these scores, we also increment a distinct word count, to be included in the log-linear model, for a total of six features.

Note that, while in a well-formed tree only one root can exist, we might end up with a forest rather than a single tree if several branches cannot be connected properly. In this case, we compute the scores on each resulting (partial) tree but treat them as if they were computed on a single tree.

5. Experiments

In this section, we empirically evaluate various aspects of string-to-dependency hierarchical machine translation, namely building the dependency tree during decoding, building it using a parser during reranking, hard restrictions vs. soft features during dependency tree construction, the de-

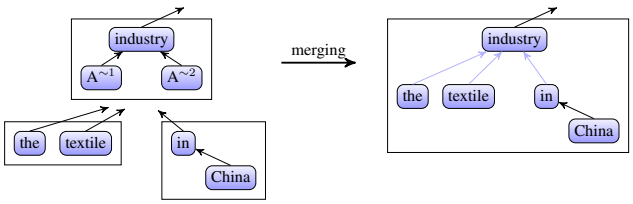


Figure 5: Merging two phrases without merging errors. All dependency pointers point into the same directions as the parent-dependencies.

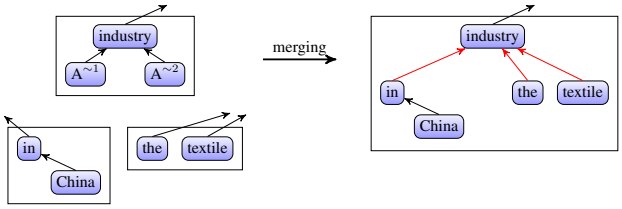


Figure 6: Merging two phrases with one left and two right merging errors. The dependency pointers point into other directions as the parent-dependencies.

pendency language model order, and the influence of features created during the dependency tree construction. Significance levels are annotated with (*) for $p < .1$ and with (**) for $p < .05$.

We employ RWTH's freely available machine translation toolkit Jane [14], a hierarchical phrase-based translation system comparable to David Chiang's Hiero [1]. The baseline setup, which is kept constant for all experiments with the same language pair, consists of the following features: 4-gram language model, phrase translation probabilities (target to source and source to target), word translation lexicons, word penalty, phrase penalty, binary markers for hierarchical phrases and generic glue rules. The word alignments have been computed based on the IBM Models with GIZA++ [15].

5.1. Chinese–English

Large parts of our experiments are carried out on the NIST Chinese–English task with around 3 million parallel training sentences and 81 million running words on the target side. The NIST 2006 evaluation set (nist06) is used as a development corpus, the NIST 2008 and 2005 sets (nist08, nist05) and a concatenation of the NIST 2002 and 2004 sets (nist0204) are used as test sets. We rely on the Stanford Dependency Parser [16] to create the dependency trees during training and reranking. The parser model was trained on the Wall Street Journal corpus.

5.1.1. Parsing Dependency Tree vs. Building During Decoding

First, we wanted to check whether dependency tree construction during decoding is more reliable than obtaining dependency trees by conducting dependency parsing directly on output hypotheses. We carried out a reranking experiment on 4000-best lists. While computationally expensive, parsing the entries of n-best lists has the advantage that dealing with merging errors or invalid phrases is not necessary. However, the tree obtained from a parse may contain even more erroneous dependencies since hypotheses are often grammatically unshaped.

We distinguish between the following feature sets:

Parse Tree dependency language model features based on a dependency tree derived from parsing the n-best hypotheses

Constructed Tree dependency language model features based on a dependency tree built during decoding

Tree Features penalty features for construction errors of the dependency tree built during decoding

Combinations of these are inspected, too. The results in Table 2 suggest that both the dependency tree derived from parsing the n-best lists as well as the tree built during decoding comparably improve the translation result on nist0204 and nist05, whereas nist08 is not that much affected. While not significant, it still seems safe to assume that strong results are achieved for a combination of constructed tree and tree features, since this can be observed for all test sets.

5.1.2. Dependency Language Model in Reranking vs. Integration into the Decoder

As the previous experiments relied on reranking only, which is unsatisfactory since search errors cannot be corrected if pruned at earlier stages, we implemented dependency LM scoring during decoding. The improvements of an integrated dependency LM scoring over 4000-best list reranking are presented in Table 3.

The BLEU scores improve on all test sets, and are now significant over the baseline for nist0204. The improvements in TER are significant for all sets.

5.1.3. Features vs. Restrictions

So far, we penalized invalid phrases but kept them in the phrase table. Shen et al. report a deterioration when the invalid phrases are filtered out of the table, which is however counteracted by the dependency language model which then raises the performance well over the baseline.

In this experiment, we analyze in which way a filtering of invalid dependency structures affects our translation performance. A smaller phrase table promises faster translation with less memory usage requirements, but possibly worse results. Table 4 shows the performance of the filtered phrase table (only valid phrases) in comparison to the unfiltered table (all phrases) and the baseline. Restricting the translation to use only valid phrases resulted in a decoding speed-up of roughly 30%, but a slightly worse performance in terms of

Table 2: Building dependency tree during decoding vs. parsing the hypotheses, on the Chinese–English task (truecase). In this experiment, the dependency LM is applied in an n-best reranking framework.

Setting	nist06 (dev)		nist0204		nist05		nist08	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
baseline	32.6	62.5	34.0	61.8	32.0	62.8	25.5	67.5
parse tree	33.0	62.2	34.4	61.6	32.6	62.2	25.2	67.2
constructed tree	33.0	62.5	34.1	61.9	32.4	62.6	25.5	67.5
constructed tree + parse tree	33.1	62.2	34.1	61.7	32.2	62.5	25.3	67.3
tree features	33.3	62.1	34.1	61.8	32.3	62.8	25.5	67.4
tree features + parse tree	33.1	62.3	34.2	61.9	32.6	62.8	25.1	67.7
tree features + constructed tree	33.3	61.7**	34.6	60.9**	32.6	62.0*	25.6	66.7**
tree features + constructed tree + parse tree	33.1	62.0	34.4	61.5*	32.4	62.6	25.4	67.2

Table 3: Dependency LM applied in reranking vs. integrated into decoding, on the Chinese–English task (truecase). Reranking is done using the constructed tree with tree features.

Setting	nist06 (dev)		nist0204		nist05		nist08	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
baseline	32.6	62.5	34.0	61.8	32.0	62.8	25.5	67.5
reranking LM	33.3	61.7**	34.6	60.9**	32.6	62.0*	25.6	66.7**
integrated LM	33.5**	60.8**	34.9**	60.3**	32.9	61.1**	26.0	65.7**

Table 4: Hard restrictions vs. features, on the Chinese–English task (truecase).

Setting	nist06 (dev)		nist0204		nist05		nist08	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
baseline	32.6	62.5	34.0	61.8	32.0	62.8	25.5	67.5
only valid phrases	32.8	62.0*	34.5	61.2	32.4	62.0*	25.4	67.1**
no merging errors	32.5	61.5**	33.8	60.9**	31.7	62.3	25.5	66.4**
all phrases	33.5**	60.8**	34.9**	60.3**	32.9	61.1**	26.0	65.7**

Table 5: Separate effect of dependency tree building features and dependency LM, on the Chinese–English task (truecase).

Setting	nist06 (dev)		nist0204		nist05		nist08	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
baseline	32.6	62.5	34.0	61.8	32.0	62.8	25.5	67.5
tree features	32.7	61.4**	34.7**	60.7**	32.5	61.5**	25.5	66.6**
dependency lm	32.9	62.3	34.4	61.4*	32.4	62.6	25.4	67.2
tree features + dependency lm	33.5**	60.8**	34.9**	60.3**	32.9	61.1**	26.0	65.7**

Table 6: Effect of dependency LM order, on the Chinese–English task (truecase).

Setting	nist06 (dev)		nist0204		nist05		nist08	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
baseline	32.6	62.5	34.0	61.8	32.0	62.8	25.5	67.5
tree features, no dependency lm	32.7	61.4**	34.7**	60.7**	32.5	61.5**	25.5	66.6**
2-gram dependency lm	33.4*	61.0**	34.8**	60.4**	32.8	61.3**	25.9	66.2**
3-gram dependency lm	33.5**	60.8**	34.9**	60.3**	32.9	61.1**	26.0	65.7**
4-gram dependency lm	33.5*	61.2**	34.9**	60.3**	33.1**	61.2**	25.9	66.2**

Table 7: Experimental results with dependency features and dependency LM, on the German–French task (truecase).

Setting	dev		test		eval10	
	BLEU	TER	BLEU	TER	BLEU	TER
baseline	20.8	67.6	21.0	67.3	36.2	53.1
rescoring tree features + dependency lm	21.1	67.0*	21.2	66.8	36.2	52.9
tree features + rescoring dependency lm	21.0	67.0*	21.3	66.6*	36.3	52.6
tree features + dependency lm	21.2	66.7**	21.6*	66.1**	36.3	52.3

BLEU and TER compared to the unrestricted translation. The restricted system still outperforms the baseline.

We also examine whether penalizing merging errors is better than completely forbidding them (no merging errors). The results show a drop in translation performance, as well. It seems that the benefits of an enforced well-formedness of constructed dependency trees do not outweigh the negative effect of restricting the decoder.

5.1.4. Effect of Tree Building Features

We now wanted to separate the effect of the dependency tree building features from the effect of the dependency language model. The results of this experiment are given in Table 5.

Both of the feature sets show improvements over the baseline for all test sets except nist08 when applied in isolation. Using the combination of dependency language model and dependency tree building features results in the best performance, which indicates that they complement one another.

5.1.5. Dependency Language Model Order

We employed trigram language models for the left-hand side and the right-hand side dependency levels in all previous experiments. If we look into the dependency trees on the training data, we realize that few words conjoin more than 2-3 dependencies on either side (see Table 8). In the training process for a dependency LM of higher order, n-grams with $n \geq 4$ would thus rarely be encountered.

We decided to experimentally affirm our assumption that a trigram is a good choice. In the experiment presented in Table 6, we compare the baseline with no language model scores and an enhanced baseline using only tree building features with setups employing bigram, trigram and 4-gram dependency LMs. The performance differences between the various language model orders seem to be negligible, except for a small increase moving from a bigram language model to a trigram language model.

Table 8: Elements per dependency layer for the English side of the Chinese–English task.

	1 element	2 elements	3 elements	≥ 4 elements
left	20 602 300	9 329 134	5 967 009	2 101 633
right	24 654 584	5 273 208	1 941 730	840 557

5.2. German–French

It is perhaps no coincidence that many syntactically motivated papers focus on a Chinese–English task since the grammar structure is quite different and additional linguistic knowledge often helps the system to improve over the baseline. To examine if we can also obtain improvements on other language pairs, we tested the setup that proved best on the NIST Chinese–English task also on the German–French language pair.

We work on the German–French translation task as defined within the Quaero project. Our parallel training corpus consists of 2 million sentences. Since the Stanford dependency parser does not provide a pre-trained model to parse French, we used the Berkeley dependency parser [17] in-

stead.

The translation results are presented in Table 7. While the methods show little impact on the eval set of 2010, the translation quality on the other test set significantly improves in both BLEU and TER when applying tree building features and dependency LM directly in decoding. Even though the performance is not improved on all test sets for the German–French task, we still consider string-to-dependency extensions to be a valuable addition to hierarchical systems even for closer-related language pairs.

6. Conclusion

We have shown that information derived from dependencies can significantly improve the translation performance on both a Chinese–English and a German–French task. By focussing on the individual aspects of the dependency features, we were able to dissect the influences that contribute to this improvement. It seems that parsing the output as compared to constructing dependency trees from structures extracted from parsed training material yields no positive impact on the result. Utilizing a dependency language model during decoding produces better results than when employed in a reranking step. Additionally, in a non-restrictive dependency tree construction process, tree building features give valuable information that can guide the translation to a grammatically more sound direction. The n-gram order of the dependency language model seems to have only a marginal influence. More interestingly, the translation performance is better whenever the decoder search space is not restricted. Even if merging errors are to be expected, it seems to be more important to offer these informations as soft features rather than to exclude certain phrases.

7. Acknowledgments

This work was partly achieved as part of the Quaero Programme, funded by OSEO, French State agency for innovation, and partly funded by the European Union under the FP7 project T4ME Net, Contract No. 249119.

8. References

[1] D. Chiang, "Hierarchical Phrase-Based Translation," *Computational Linguistics*, vol. 33, no. 2, pp. 201–228, June 2007.

[2] L. Shen, J. Xu, and R. Weischedel, "A New String-to-Dependency Machine Translation Algorithm with a Target Dependency Language Model," in *Proc. of the ACL/HLT*, Columbus, Ohio, June 2008, pp. 577–585.

[3] ——, "String-to-Dependency Statistical Machine Translation," *Computational Linguistics*, vol. 36, no. 4, pp. 649–671, Dec. 2010.

[4] D. Stein, S. Peitz, D. Vilar, and H. Ney, "A Cocktail of Deep Syntactic Features for Hierarchical Machine Translation," in *Proc. of the AMTA*, Denver, CO, Oct./Nov. 2010.

[5] M. Huck, J. Wuebker, C. Schmidt, M. Freitag, S. Peitz, D. Stein, A. Dagnelies, S. Mansour, G. Leusch, and H. Ney, "The RWTH Aachen Machine Translation System for WMT 2011," in *Proc. of the EMNLP/WMT*, Edinburgh, UK, July 2011, pp. 405–412.

[6] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a Method for Automatic Evaluation of Machine Translation," in *Proc. of the ACL*, Philadelphia, PA, July 2002, pp. 311–318.

[7] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul, "A Study of Translation Edit Rate with Targeted Human Annotation," in *Proc. of the AMTA*, Cambridge, MA, Aug. 2006, pp. 223–231.

[8] M. Galley and C. D. Manning, "Quadratic-time Dependency Parsing for Machine Translation," in *Proc. of the ACL/AFNLP*, vol. 2, 2009, pp. 773–781.

[9] N. Bach, Q. Gao, and S. Vogel, "Source-side Dependency Tree Reordering Models with Subtree Movements and Constraints," in *Proc. of the MTSummit-XII*, Ottawa, Canada, August 2009.

[10] Y. Gao, P. Koehn, and A. Birch, "Soft Dependency Constraints for Reordering in Hierarchical Phrase-Based Translation," in *Proc. of the EMNLP*, Edinburgh, Scotland, UK., July 2011, pp. 857–868.

[11] J. Xie, H. Mi, and Q. Liu, "A Novel Dependency-to-string Model for Statistical Machine Translation," in *Proc. of the EMNLP*, Edinburgh, Scotland, UK., July 2011, pp. 216–226.

[12] C. Quirk and A. Menezes, "Dependency treelet translation: the convergence of statistical and example-based machine-translation?" *Machine Translation*, vol. 20, no. 1, pp. 43–65, 2006.

[13] M.-C. de Marnee and C. D. Manning, "Stanford typed dependencies manual," 2008.

[14] D. Vilar, D. Stein, M. Huck, and H. Ney, "Jane: Open Source Hierarchical Translation, Extended with Reordering and Lexicon Models," in *Proc. of the ACL/WMT*, Uppsala, Sweden, July 2010, pp. 262–270.

[15] F. J. Och and H. Ney, "A Systematic Comparison of Various Statistical Alignment Models," *Computational Linguistics*, vol. 29, no. 1, pp. 19–51, Mar. 2003.

[16] D. Klein and C. D. Manning, "Accurate Unlexicalized Parsing," in *Proc. of the ACL*, Sapporo, Japan, July 2003, pp. 423–430.

[17] S. Petrov, L. Barrett, R. Thibaux, and D. Klein, “Learning Accurate, Compact, and Interpretable Tree Annotation,” in *Proc. of the ACL*, Sydney, Australia, July 2006, pp. 433–440.