Sentence Realization with Unlexicalized Tree Linearization Grammars

Rui WANG Yi ZHANG
DFKI GmbH, Germany
{ruiwang,yizhang}@dfki.de

Abstract
Sentence realization, as one of the important components in natural language generation, has taken a statistical swing in recent years. While most previous approaches make heavy usage of lexical information in terms of N-gram language models, we propose a novel method based on unlexicalized tree linearization grammars. We formally define the grammar representation and demonstrate learning from either treebanks with gold-standard annotations, or automatically parsed corpora. For the testing phase, we present a linear time deterministic algorithm to obtain the 1-best word order and further extend it to perform *exact* search for n-best linearizations. We carry out experiments on various languages and report state-of-the-art performance. In addition, we discuss the advantages of our method on both empirical aspects and its linguistic interpretability.

Keywords: Tree Linearization Grammar, Sentence Realization, Dependency Tree.

Proceedings of COLING 2012: Posters, pages 1301–1310,
COLING 2012, Mumbai, December 2012.

1 Introduction

Natural language generation (NLG) is the key processing task of producing natural language from some level of abstract representation, either syntactic or (more often) semantic. The long-standing research in NLG has gone through in-depth investigation into various sub-steps, including content planning, document structuring, lexical choices, surface realization, etc. The traditional generation systems typically rely on a rich set of annotation as input and is tightened to specific frameworks or internal representation, making the reuse of other natural language processing components difficult. Inspired by the successful application of statistical methods in natural language analysis, researchers shifted towards using standardized linguistic annotations to learn generation models. In particular, the now ever-so-popular dependency representation for syntacto-semantic structures has made its way into the sentence realization task, as evident by the recent Generation Challenge 2011 Surface Realization Shared Task (Belz et al., 2011). Given the full-connectedness of the input structure, the task of surface realization concerns mainly about the linearization process[1], which shall determine the ordering of words in the dependency structures.

While the earlier work like Langkilde and Knight (1998) showed that the N-gram language models can work well on the tree linearization, more recent study shows that improvements can be achieved by combining the language model outputs with discriminative classifiers (Filippova and Strube, 2009; Bohnet et al., 2010). On the other hand, we see that relatively few results have been reported on a grammar-based approach, where linearization rules are used instead to determine the word order within a given structured input.

In this paper, we use **tree linearization grammar**s to specify the local linearization constraints in bilexical single-headed dependency trees. Unlexicalized linearization rules and their probabilities can be learned easily from the treebank. By using a dependency parser, we expand the grammar extraction to **automatically parsed** dependency structures as well. The linearization model is **fully generative**, which can produce N-best word orders for each dependency input. Detailed evaluation and error analysis on **multiple languages** show that our grammar-based linearization approach achieves state-of-the-art performance without language specific tuning or detailed feature engineering. The resulting linearization rules are comprehensible and reflect linguistic intuitions.

2 Unlexicalized Tree Linearization Grammar

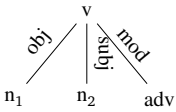
The task of tree linearization takes the unordered single-headed bilexical dependency tree as input, and produces the surface sentence with determined word order. We define the tree linearization grammar to be a set of rules that licenses the legitimate linearization of the tree.

More specifically, we define a *local configuration* $\mathscr{C} = \langle w_0, \{\langle r_1, w_1\rangle, \cdots, \langle r_n, w_n\rangle\}\rangle$ to be an unordered dependency tree of height 1, with w_0 as the *head* , and $\{w_1, w_2, \cdots, w_n\}$ as the immediate *dependents* (daughters) with corresponding dependency relations $\{r_1, r_2, \cdots, r_n\}$. A linearization rule $\mathscr{L}$ is defined to be:

$$\mathscr{L} : \mathscr{C} \Rightarrow \langle w'_0, w'_1, \cdots, w'_n\rangle \quad s.t. \quad \begin{array}{l} \forall i, 0 \le i \le n, w'_i \in \{w_0, \cdots, w_n\} \\ \forall i, j, 0 \le i, j \le n, w'_i = w'_j \text{iff } i = j \end{array} \tag{1}$$

which determines the complete order of all the words on the LHS of the local configuration. For an unlexicalized tree linearization rule, w_i are syntactic categories instead of words. In our later experiments, we use either coarse- or fine-grained parts-of-speech as representation of words in the configurations. Below is an example local configuration with two alternative linearization rules:

[1]For morphologically-rich languages, an additional inflection realizer is needed as a postprocessor.



$\Rightarrow \langle n_2, adv, v, n_1 \rangle$
$\Rightarrow \langle n_2, v, n_1, adv \rangle$

Assuming the projectivity of the dependency structure, we can find the linearization of the complete sentence if the linearization of each local configuration is determined by one of the rules. In practice, the linearization of many local configurations are ambiguous. We define a probabilistic tree linearization grammar by attaching a conditional probability distribution to the rules:

$$\mathbf{Pr} : \mathscr{L} \rightarrow [0,1] s.t. \forall \mathscr{C} \in \mathscr{C}, \sum_{\forall \mathscr{L} \in \mathscr{L}, LHS(\mathscr{L}) = \mathscr{C}} \Pr(\mathscr{L}) = 1 \quad (2)$$

where $\mathscr{C}$ is the set of all local configurations, and $\mathscr{L}$ the set of all linearization rules in the grammar. The probability of a sentence linearization given an input dependency structure $\mathscr{D}$ is then defined as: $P(\mathscr{L}_{\mathscr{D}}) = \prod_{\mathscr{L} \in \mathscr{L}_{\mathscr{D}}} \Pr(\mathscr{L})$, where $\mathscr{L}_{\mathscr{D}}$ is the linearization of the complete sentence with the application of one linearization rule $\mathscr{L}$ on each local configuration in the input $\mathscr{D}$.

Note that although we assume the projectivity of the dependency structure in this paper, it is possible to extend the definition of the tree linearization grammar to also encode the discontinuities in the syntactic structure (e.g., by explicitly marking the gaps in the structure and pointers to their fillers). Thorough investigation in this direction belongs to our future work. Nevertheless, empirical results from section 4 suggest that non-projectivity is not a major source of errors for the languages and datasets used in our experiments.

Similar to the treebank-based approach to grammar extraction for parsing, we extract linearization rules from the annotated dependency treebank with determined word order. Each local configuration and its linearization is then gathered as a rule. Due to the unlexicalized approach we take, this produces a relatively small grammar which can be manually interpreted. To estimate the rule probalilities **Pr**, we simply use the maximum likelihood estimation: $\Pr(\mathscr{L}) = \frac{Freq(\mathscr{L})}{Freq(LHS(\mathscr{L}))}$.

3 N-Best Tree Linearization

We start with the grammar-based deterministic tree linearization algorithm which outputs 1-best linearization by recursively finding the best linearization for each local configuration. The time complexity of the algorithm is $\mathscr{O}(n)$, where n is the number of words in the dependency tree.

The n-best linearization algorithm is an extension to the 1-best procedure. Each `hypothesis` represents a state in the search for the n-best linearizations for the sub-tree under a given `node`. It further relies on a vector `indices`, where `indices[0]` identifies the index of the linearization rule, and the remaining elements `indices[1..k]` point to the sub-states in the n-best linearization of the dependents. Top level procedure `linearize-node(root,n)` will iteratively instantiate the top-n linearization hypothesis of the root. The main procedure `hypothesize-node(node, i)` creates the i^{th} best hypothesis of `node`, which recursively finds the linearization of the sub-trees.

According to our definition, the linearization probability of a given node n can be calculated by multiplying the rule probability with the sub-linearization probabilities of the dependents:

$$P(\mathscr{L}_n) = \Pr(rule(\mathscr{L}_n)) * \prod_{d \in dependents(n)} P(\mathscr{L}_d) \quad (3)$$

```
1  procedure linearize-node(root, n) ≡
2    results ← ⟨⟩; i ← 0;
3    do
4      hypothesis ← hypothesize-node(root, i); i ← i + 1;
5      if (not hypothesis) break;
6      new ← instantiate-hypothesis(hypothesis);
7      n ← n − 1; results ← results ⊕ ⟨new⟩;
8    while (n ≥ 1);
9    return results;

10 procedure hypothesize-node(node, i) ≡
11   if (node.hypotheses[i]) return node.hypotheses[i];
12   if (i = 0) then
13     node.rules ←sorted-rules(node.config); indices ← ⟨0⟩
14     daughters ← ⟨⟩;
15     for each (dnode in node.dependents) do
16       daughters ← daughters ⊕ ⟨hypothesize-node(dnode, 0)⟩;
17       indices ← indices ⊕ ⟨0⟩;
18     new-hypothesis(node, daughters, indices);
19   if (hypothesis ← node.agenda.pop()) then
20     for each (indices in advance-indices(hypothesis.indices)) do
21       daughters ← ⟨⟩;
22       for each (dnode in node.dependents) each (j in indices[1..k]) do
23         daughter ← hypothesize-node(dnode, j);
24         if (not daughter) then daughters ← ⟨⟩; break
25         daughters ← daughters ⊕ ⟨daughter⟩;
26       if (daughters) then new-hypothesis(node, daughters, indices)
27     node.hypotheses[i] ← hypothesis;
28     return hypothesis;

29 procedure new-hypothesis(node, daughters, indices) ≡
30   hypothesis ← new hypothesis(node, daughters, indices);
31   node.agenda.insert(score-hypothesis(hypothesis), hypothesis);
```

Figure 1: N-best tree linearization algorithm

This calculation is achieved within the procedure `score-hypothesis`. Since the n^{th} best linearization must be different from one of the top $n-1$ linearizations in just one position (either one of the dependents' sub-linearization or the linearization rule chosen for the top node), `advance-indices` will find all such possibilities. With such *lazy* expansion of the search frontier, only the immediate candidates are added to the local agenda of each node.

4 Experiments

For the evaluation, we use the dependency treebanks for multiple languages from the CoNLL-shared task 2009[2] (Hajič et al., 2009). Additional unlabeled English texts from L.A. Times & Washington Post of the North American News Text (NANC) (Supplement)[3] are used for training the English models. Testing results are reported on the development sets of the CoNLL dependency treebanks. In addition to the automatic metrics such as BLEU (Papineni et al., 2002) and Ulam's distance (Birch et al., 2010), we also manually evaluate the quality of the system outputs (Section 4.3).

4.1 Basic Models

For the basic models, we compare our grammar-based approaches with three baselines, Random, N-Gram, and Rank. The first baseline simply produces a random order of the words; the second model can be viewed as a simplified version of (Guo et al., 2011)'s basic model[4]; and the third model

[2] `http://ufal.mff.cuni.cz/conll2009-st/index.html`

[3] `http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC98T30`

[4] Instead of using grammatical functions derived from lexical functional grammar (LFG), we use the dependency relation and the parts-of-speech as our syntactic categories. For the previous example, we obtain N-gram counts from

is a log-linear model, which is trained on each word's relative position in its local configuration[5]. For the main approach based on linearization grammars, we have two configurations using either coarse- or fine-grained part-of-speech (CPOS vs. POS)[6]. Notice that the baseline system N-Gram can also choose between CPOS and POS.

In Table 1, 'Covered' rows report the results on the subcorpus whose sentences are fully covered by the grammar, and 'Overall' rows report the results on the complete test corpus with Rank baseline as the backoff model for the out-of-grammar configurations. As Rank can only produce 1-best linearization, we set the score for that configuration as 1 for further calculation in Equation (3). '1-best' is the deterministic tree linearization result, while 'upper bound (1000)' gives the upper bound of n-best linearization with $n = 1000$.

Models		POS	CPOS	Baselines		
				Rank	N-Gram	Random
Coverage	Sent. (1334)	451 (33.8%)	711 (53.3%)	-	-	-
	Config. (17282)	15843 (91.7%)	16423 (95.0%)			
BLEU						
Covered	1-best	92.65	90.64	-	-	-
	upper bound (1000)	96.31	95.31			
Overall	1-best	81.63	**83.28**	73.09	43.22	32.34
	upper bound (1000)	84.08	87.13	-	66.55	44.90

Table 1: Performance of the basic models

We observe that although the grammar with fine-grained POS achieves better performance on the data covered by the grammar, the coverage is relatively low. On the overall results, when combined with the Rank backoff model, the CPOS model achieves a good balance between 'precision' and 'recall', and also outperforms all the baselines with large margins (10+ BLEU points). We will refer to this system as our Base model for the rest of this section.

While the configuration level coverage is over 95%, the Base grammar only achieves full coverage on 53% of the sentences. We investigate the possible ways of expanding the coverage of the linearization grammar in Section 4.2. Also, when comparing the Base model with the n-best upper bound, the difference of 4 BLEU points suggests that a better ranking model can potentially achieve further improvements on the linearization. This will be discussed in Section 4.3.

4.2 Experiments with Automatically Parsed Data

Self-training has been shown to be effective for parser training (McClosky et al., 2006). It expands the training observations on new texts with hypothesized annotation produced by a base model. In our case, we can obtain further linearization observations from unannotated sentences, and rely on a parser to produce the dependency structures[7].

We use a state-of-the-art dependency parser, MSTParser (McDonald et al., 2005), and train it with the same data with gold-standard dependency annotations using the second order features and a projective decoder. For the additional data, we use a fragment of the NANC corpus (765670

sequences $\langle n|subj, adv|mod, v|hd, n_1|obj\rangle$ and $\langle n_2|subj, v|hd, n_1|obj, adv|mod\rangle$. On top of such instances from all the configurations, we train a tri-gram model.

[5]The features we use include token features, lemma and part-of-speech, and the dependency relation. We differentiate parent and children nodes by adding different prefixes.

[6]In the CoNLL data, the coarse-grained POS is the first character of the fine-grained POS.

[7]Unlike parser self-training, we do not update the parsing model, but expect the extra observations to help improve the coverage of the linearization grammar.

sentences in total). The 1-best linearization with the additional corpus improved from 83.28 to 83.94 BLEU, with the oracle (1000) upper-bound improved from 87.13 to 88.82 BLEU.

Although the performance on the grammar-covered sentences drops slightly, the overall performance improves steadily for both 1-best and the upper bound. We also notice that on such high range BLEU scores (above 80), the differences are less indicative of the actual quality of the linearization. We will address this issue in the next section.

4.3 Manual Analysis

In the previous experiments, we have observed the performance difference between the 1-best result and the n-best upper bound. In an attempt to improve the selection of the best linearization, we incorporate a simple tri-gram language model at the surface level to re-rank the n-best output of the Base model (LM-Rerank), and hope it will compensate for the lack of lexical information in our unlexicalized linearization grammar. We train the language model on the same data and choose $n = 1000$.

When we perform a pair-wise comparison of the output from these two systems, the results show that in 28% of the cases Base is better; and only in 14% of the cases LM-Rerank is better. To further investigate the difference between the two systems, we carry out a manual analysis on two aspects: 1) *comprehensiveness* and 2) *grammaticality*, which are similar to the measurements used in the surface realization evaluation (Belz et al., 2011). In particular, we have three levels of judgement for both criteria, ranging from 0 to 2. As both systems output exactly the same gold standard linearizations in almost half of the cases, we calculate the BLEU score for each sentence and randomly sample 100 sentences between the range $(75.0, 90.0]$. This allows us to 'zoom in' on the (error) characteristics of the two systems.

Each sentence from both systems is annotated by two annotators on two criteria with reference to the gold standard linearization. For both criteria, the annotations are mostly agreed, with Cohen's Kappa scores (Cohen, 1960) $\kappa = 0.83$ for *comprehensiveness* and $\kappa = 0.87$ for *grammaticality*. Table 2 shows the results only on the agreed cases.

	Comprehensiveness	*Grammaticality*	Perfect
Base	84.1%	**77.1%**	28.8%
LM-Rerank	**90.1%**	73.2%	36.7%

Table 2: Agreed manual evaluation results on sentences within BLEU range $(75.0, 90.0]$

We sum up the scores for both criteria separately, and divide by a sum of maximal scores. The 'Perfect' column indicates the portion of sentences which get full scores for both criteria, i.e., they are correct and fluent, though being different from the gold standard.

Notice that the sampled sentences do not reflect the performance of two configurations as a whole due to the selection process. However, the differences on two criteria reflect different characteristics of the two systems. Base tends to output grammatical linearization, though the results could be less fluent and hard to comprehend; LM-Rerank is more fluent and comprehensible, though might violate grammaticality. Furthermore, the result indicates that within this BLEU range about 1/3 of the output sentences are perfect linearization, although the BLEU scores are not 1. We view this issue as the inadequacy of 1-best evaluation for this task, as among the n-best output, we have observed more than one correct realizations. However, proposing a better evaluation method is out of the scope of this paper, which we will leave for our future work.

Sentences
Gold: ***[" The market is overvalued , not cheap , " says]*** *Alan Gaines of the New York money - management firm Gaines Berland .* System: *Alan Gaines of the New York money - management firm Gaines Berland* ***[says , " The market is overvalued , not cheap . "]***
Gold: *... than many taxpayers working at the same kinds of jobs and* ***[perhaps]*** *supporting families .* System: *... than many taxpayers* ***[perhaps]*** *working at the same kinds of jobs and supporting families .*
Gold: *... to set* ***[aside]*** *provisions covering all its C$ 1.17 billion in non - Mexican LDC debt .* System: *... to set provisions covering all C$ its 1.17 billion in non - Mexican LDC debt* ***[aside]*** *.*
Gold: *Good service programs require recruitment , screening , training and supervision –* ***[all of high quality]*** *.* System: ***[all of high quality]*** *– Good service programs require recruitment , screening , training and supervision .*

Table 3: Examples of the system output compared with the gold standard

We list several examples of the system output in Table 3. One major source of errors is the clustering of punctuations, in particular, commas, as they are not differentiable at the configuration level for the backoff model Rank. This occurs less with the LM-Rerank model. The free movement of modifiers (adjectives, adverbs, modifying prepositional phrases, etc.) poses a serious challenge for automatic evaluation, as in most cases the meaning does not change. However, in the second example in the table, due to the coordinate structure, the movement of *"perhaps"* does change the meaning of the sentence. Furthermore, the context-freeness of the linearization rules do not concern the *'heaviness'* of the dependent NP, hence (wrongly) preferring the unnatural placement of *"aside"* to the end of the sentence in the third example. The last example shows that even when the generated sentence is perfectly grammatical, the discourse semantics could change drastically.

4.4 Multilinguality

To investigate the multilingual applicability of our approach, we further experiment with five more languages: Catalan (CA), Chinese (CN), Czech (CZ), German (DE), and Spanish (ES). There is no language-specific tuning, so this is achieved easily with the availability of the CoNLL 2009 Shared Task datasets. We show some basic statistics of the datasets in Table 4 as well as the system performance under two automatic measurements: BLEU and Ulam's distance. The latter is the minimum number of single item movements of arbitrary length required to transform one permutation into another (Ulam, 1972), which is the same as the 'di' measurement used by Bohnet et al. (2010) and others.

Languages		CA	CN	CZ	EN	DE	ES
No. of CPOS Tag		12	13	12	24	10	12
Avg. Token / Sent.		31.0	30.0	16.8	25.0	16.0	30.4
Grammar							
Avg. Config. / Sent.		13.1	14.0	8.3	12.4	6.0	13.2
Coverage	Sent.	578 / 1724 (33.5%)	790 / 1762 (44.8%)	498 / 5228 (9.5%)	724 / 1334 (54.3%)	1512 / 2000 (**75.6%**)	650 / 1655 (39.3%)
	Config.	22526 / 24546 (91.8%)	24749 / 26250 (94.3%)	43552 / 49751 (87.5%)	16536 / 17369 (95.2%)	11925 / 12503 (**95.4%**)	21920 / 23511 (93.2%)
BLEU							
Covered	1-best	84.51	88.67	82.00	91.95	78.52	79.93
	upper bound (1000)	91.77	94.49	93.60	96.20	88.01	89.78
Overall	1-best	75.79	**81.48**	66.59	**84.89**	73.85	73.10
	upper bound (1000)	80.61	86.52	76.85	88.75	82.09	79.75
Ulam's distance							
Covered	1-best	0.890	0.946	0.867	0.950	0.857	0.871
	upper bound (1000)	0.949	0.973	0.965	0.978	0.934	0.941
Overall	1-best	0.838	**0.891**	0.771	**0.911**	0.829	0.820
	upper bound (1000)	0.875	0.914	0.856	0.934	0.897	0.869

Table 4: Performance of the multilingual models

Notice that the best coverage of the grammar is on the German data, which is mainly due to the short average sentence length (16.0 tokens / sentence) and the flatness of the tree (6.0 configura-

tions / sentence). However, the high coverage does not guarantee good performance, as for each configuration, the linearization selection could still be ambiguous. Overall, the best performances come from English and Chinese, whose word orders are relatively strict; while Czech has the worst performance due to its relatively free word order, and the coverage of the grammar is also the lowest.

We observe 803 non-projective inputs from the Czech test set (15.4%), and 106 sentences from German (5.3%); for the other languages, almost all the trees are projective. The proposal of Bohnet et al. (2012) to use a separate classifier to predict the lifting of non-projective edges in a dependency tree can be combined with the use of linearization rules in our approach in the future.

Note that although we test our models on the same data source as the surface realization shared task[8], subtle differences in the preprocessing of the data and/or the evaluation scripts make the direct comparison to previously reported results difficult. Some comparison of different approaches and reported results will be discussed in the next section.

5 Discussion and Future Work

Several works on statistical surface realization have been reported recently. Ringger et al. (2004) proposed several models, and achieved 83.6 BLEU score on the same data source.They also tested their approaches on French and German data, but with predicate-argument structures as input. One of the interesting features of our approach is the generative nature of the model. Unlike the previous work of (Filippova and Strube, 2009; Bohnet et al., 2010) who relied on discriminative modeling for the selection of the realization, our approach actually produces the realization probabilities, and does not rely on ad hoc pruning of the search space. Filippova and Strube (2009) (and their previous paper) reported 0.88 Ulam's distance for English and 0.87 for German, but their evaluation is at the clause level instead of full sentences. Bohnet et al. (2010)'s experiments were also on the CoNLL datasets, achieving 85 BLEU score for Chinese, 89.4 for English, 73.5 for German, and 78 for Spanish. Their system was ranked the first place in the surface realization shared task, followed by Guo et al. (2011)'s dependency-based N-gram approach. The permutation filtering technique they use is essentially similar to our linearization rules. As we show in the manual evaluation, the BLEU scores are not always indicative (especially at the higher range) of the generation quality, in the future, we are interested in a more elaborate manual analysis of their results.

While we are achieving satisfying results with our method on multiple languages without language-specific tuning, it should be noted that the dependency tree linearization task is only part of the sentence realization workflow, along with other subtasks such as lexical choices, morphological realizer, etc. The linearization rules learned are not a full-fledged grammar covering the entire syntactic layer of the language, but rather the complements to the morphological or grammatical relations given by the dependency inputs and they specify the linear precedence of the words. In comparison to the other sentence realization systems which relies on richer frameworks and accept more abstract semantic inputs, our task does not touch the syntacto-semantic interface. Nevertheless, it is interesting to note that one of the key challenges in semantics-based sentence realization is the lack of word-order constraints, hence the inefficiency (Carroll et al., 1999; Carroll and Oepen, 2005; Espinosa et al., 2008). With our efficient grammar-based linearization algorithms, extra efficiency boost to the deeper generation workflow can be achieved by pruning the implausible orderings.

Acknowledgments

The work is partially supported by the *Deependance* project funded by BMBF (01IW11003).

[8] Unfortunately, we were not able to acquire the datasets after the shared task has ended.

References

Belz, A., White, M., Espinosa, D., Kow, E., Hogan, D., and Stent, A. (2011). The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France. Association for Computational Linguistics.

Birch, A., Osborne, M., and Blunsom, P. (2010). Metrics for MT evaluation: evaluating reordering. *Machine Translation*, pages 1–12.

Bohnet, B., Björkelund, A., Kuhn, J., Seeker, W., and Zarriess, S. (2012). Generating non-projective word order in statistical linearization. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 928–939, Jeju Island, Korea. Association for Computational Linguistics.

Bohnet, B., Wanner, L., Mill, S., and Burga, A. (2010). Broad coverage multilingual deep sentence generation with a stochastic multi-level realizer. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 98–106, Beijing, China.

Carroll, J., Copestake, A., Flickinger, D., and Poznanski, V. (1999). An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation*, pages 86–95, Toulouse, France.

Carroll, J. and Oepen, S. (2005). High-efficiency realization for a wide-coverage unification grammar. In Dale, R. and Wong, K. F., editors, *Proceedings of the 2nd International Joint Conference on Natural Language Processing*, pages 165–176, Jeju, Korea.

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37.

Espinosa, D., White, M., and Mehay, D. (2008). Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of ACL-08: HLT*, pages 183–191, Columbus, Ohio. Association for Computational Linguistics.

Filippova, K. and Strube, M. (2009). Tree linearization in english: Improving language model based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 225–228, Boulder, Colorado.

Guo, Y., Wang, H., and van Genabith, J. (2011). Dependency-based n-gram models for general purpose sentence realisation. *Natural Language Engineering*, 17(04):455–483.

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., Straňák, P., Surdeanu, M., Xue, N., and Zhang, Y. (2009). The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado. Association for Computational Linguistics.

Langkilde, I. and Knight, K. (1998). Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 704–710, Montreal, Quebec, Canada.

McClosky, D., Charniak, E., and Johnson, M. (2006). Effective self-training for parsing. In *Proceedings of the Conference on Human Language Technology and North American chapter of the Association for Computational Linguistics (HLT-NAACL 2006)*, Brooklyn, New York, USA. Association for Computational Linguistics.

McDonald, R., Pereira, F., Ribarov, K., and Hajic, J. (2005). Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT-EMNLP 2005*, pages 523–530.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Ringger, E., Gamon, M., Moore, R. C., Rojas, D., Smets, M., and Corston-Oliver, S. (2004). Linguistically informed statistical models of constituent structure for ordering in sentence realization. In *Proceedings of Coling 2004*, pages 673–679, Geneva, Switzerland. COLING.

Ulam, S. M. (1972). Some ideas and prospects in biomathematics. *Annual Review of Biophysics and Bioengineering*, pages 277–292.