An overview of Amalgam: A machine-learned generation module

Simon Corston-Oliver, Michael Gamon, Eric Ringger, Robert Moore
Microsoft Research
Redmond, WA 98052
{simonco, mgamon, ringger, bobmoore}@microsoft.com

Abstract

We present an overview of Amalgam, a sentence realization module that combines machine-learned and knowledge-engineered components to produce natural language sentences from logical form inputs. We describe the decomposition of the task of sentence realization into a linguistically informed series of steps, with particular attention to the linguistic issues that arise in German. We report on the evaluation of component steps and of the overall system.

1 Introduction

Since the mid 1990s, there has been increasing interest in the application of statistical and machine learning techniques to various aspects of natural language generation, ranging from learning plans for high-level organization of texts and dialogues (Zukerman et al., 1998; Duboue and McKeown, 2001) or ensuring that the macro properties of generated texts such as the distribution of sentence lengths and lexical variety mirror the properties of naturally occurring texts (Oberlander and Brew, 2000) to sentence planning (Walker et al., 2001.). As generation proceeds through successively less abstract stages, nearing the final output string, it would appear that current generation systems are still likely to employ knowledge-engineered approaches. Indeed Walker et al. (2001), commenting on sentence realization, rather succinctly summarize what is no doubt a widely-held belief, namely that "This phase is not a planning phase in that it only executes decisions made previously." Although the kinds and number of decisions to be made during sentence realization will depend on the nature of the prior sentence planning phase and on the linguistic complexity of the domain, myriad encoding decisions must still be made. Machine learning approaches have been successfully applied to such aspects of sentence realization as determining the appropriate form of referring expressions (Poesio et al., 1999) and the grammatical relations of noun phrases (Corston-Oliver, 2000) as well as performing lexical selection (Bangalore and Rambow, 2000b). Traditional knowledge engineering approaches to sentence realization founder on our inadequate understanding of the mapping from propositional content to surface form, a mapping that encompasses such problems as equi-NP deletion, movement operations such as extraposition and left-dislocation, ordering of modifiers within constituents, and voice alternations. Research in knowledge engineered solutions to these problems continues (see for example, Aikawa et al., 2001). However, the task of broad-coverage sentence realization would appear to be of comparable complexity to sentence analysis, and equally difficult to adapt to new domains. Sentence realization therefore appears to be an ideal candidate for statistical and machine-learned approaches.

Some recently described systems have attempted to side-step the encoding decisions involved in sentence realization by proposing alternative realizations of an input semantic form and leaving it to a word-level language model to select the most likely candidate sentence for output. The Nitrogen system (Langkilde and Knight, 1998a, 1998b), for example, uses a rather permissive knowledge-engineered component to propose candidate output sentences that are then scored using word bigrams. Statistics garnered from actual texts are thus used as a substitute for deeper knowledge. The addition of syntactic information, either to constrain the range of candidate sentences or to augment the n-gram model, has produced favorable improvements over n-gram models used alone (Bangalore and Rambow, 2000a; Ratnaparkhi, 2000).

In the current paper we describe an on-going research project code-named Amalgam. Amalgam is a (predominantly) machine-learned generation module that performs sentence realization and a small degree of lexical selection. Amalgam takes as input a logical form graph. Proceeding through a series of machine-learned and knowledge-engineered steps, it transforms that graph into a fully articulated tree structure from which an output sentence can be read. Amalgam has been successfully applied to the realization of non-trivial German sentences in diverse technical domains. An extended description is given in Gamon et al. (2002a).

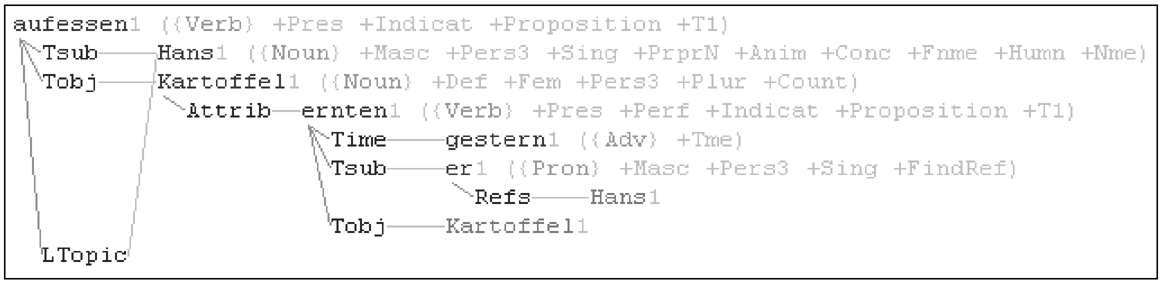
2 Linguistic issues in German generation

Although English and German are closely related languages, they now differ typologically in dramatic ways. German makes a three-way distinction in lexical gender (masculine, feminine, neuter). Nominal elements are morphologically marked for one of four grammatical cases (nominative, accusative, dative and genitive), with adjectives and determiners agreeing in gender, number and case. Verbal position is fixed, and is sensitive to clause type; the order of other constituents is relatively free. So-called "separable prefixes" are elements that may occur prefixed to verbs or as separate elements at a great distance from the verb to which they are syntactically dependent. Extraposition, a rare phenomenon in English text, is pervasive: depending on genre, one quarter to one third of all relative clauses in natural German texts are extraposed (Gamon et al., 2002b; Uszkoreit et al., 1998). Relatively free constituent order, separable prefixes and extraposition force us to deal with long distance dependencies much more frequently than in the analysis and generation of English sentences.

3 The logical form representation

The logical form representation that we employ is a graph data structure that expresses the propositional content of a sentence. Nodes in the graph contain lemmas (citation forms) of content words with additional annotations for definiteness, number, person and so on. Relations between nodes are indicated by labeled arcs. The logical form normalizes surface syntactic alternations. For example, active and passive alternations receive the same graph representation, differing only in the presence of the [+Pass] feature on the verbal head of the logical form that results from the analysis of the passive. Since the logical form is a graph, nodes may have more than one parent. Figure 1, *Hans isst die Kartoffeln auf, die er gestern geernet hat*, "Hans eats the potatoes up, which he has gathered yesterday" (i.e., "Hans eats up the potatoes which he gathered yesterday"), illustrates the graph nature of the representation.[1] A given node may be in more than one relationship to the same parent, as is the case with *Hans*, which functions both as the underlying subject of *aufessen* "to eat up" and the logical topic of the sentence. A given node may also be dependent on more than one parent node, as is the case with *Kartoffel* "potato", which functions as both the object of *aufessen* and the object of *ernten* "to gather". Long distance dependencies receive a rather direct representation in the logical form, as shown in the representation of the extraposed relative clause *die er gestern geernet hat*, "that he gathered yesterday", which is represented in the logical form as directly dependent on *Kartoffel*, which it modifies semantically.

[1] In order to simplify the visual representation of the logical form, the display attempts to minimize crossing lines. Nodes sharing the same lemma and numerical index are in fact the same node in the underlying data structures.



```
aufessen1 ({Verb} +Pres +Indicat +Proposition +T1)
  Tsub----Hans1 ({Noun} +Masc +Pers3 +Sing +PrprN +Anim +Conc +Fnme +Humn +Nme)
  Tobj----Kartoffel1 ({Noun} +Def +Fem +Pers3 +Plur +Count)
             Attrib--ernten1 ({Verb} +Pres +Perf +Indicat +Proposition +T1)
                       Time----gestern1 ({Adv} +Tme)
                       Tsub----er1 ({Pron} +Masc +Pers3 +Sing +FindRef)
                                 Refs----Hans1
                       Tobj----Kartoffel1
   LTopic
```

Figure 1 Logical form for the sentence *Hans isst die Kartoffeln auf, die er gestern geernet hat.*

4 The sentence realization process

For our experiments, we begin with a broad-coverage grammar of German (in the framework described in (Heidorn, 2000)) that parses sentences to produce a syntactic analysis. Additional processing yields a logical form analysis. Our task is essentially to learn the mapping from the output logical forms to the input sentences, with reference to the intermediate syntactic analysis. The goal is to apply Amalgam to the realization of sentences in contexts such as machine translation in which the input logical form is constructed by transfer operations. In the interim, using logical forms that result from the analysis of German text frees us from issues related to noise introduced by the process of transfer. We do not require an annotated corpus; we are able to create unlimited (though imperfect) quantities of training data by application of the German analysis system. We have been using a corpus of 100,000 German sentences drawn from technical manuals and computer help files. Depending on the phenomenon of interest, each sentence typically produces multiple training cases. For some models, such as the punctuation model, the use of the full set of sentences yielded as many as one million cases.

The first stage in the Amalgam process is a procedure that converts the input logical form into a more manageable tree data structure in which linear order has not been determined. Nodes with multiple parents in the input logical form are duplicated and co-indexed in the tree data structure. By consultation of the system dictionary, each node is annotated with lexical features such as subcategorization information that is not otherwise needed in the logical form. Nouns with nominal modifiers are collapsed to form compounds. [2]

Most of the remaining stages involve machine learning. With the exception of the generative language model used to determine intra-constituent order, all machine-learned modules have been implemented as decision tree classifiers using the WinMine toolkit (Chickering, nd.). Decision trees provide an easily accessible inventory of the selected features and some indication of their relative importance in predicting the target features in question. Significantly, decision trees built by WinMine predict a probability distribution over all possible target values. We formulate a series of questions that lend themselves to implementation in decision tree classifiers. For example, we ask, “What case should this NP have?”, “Which determiner from the set {NULL, definite determiner, indefinite determiner, …} should modify this NP?”, “Should this constituent be realized?”. Features are extracted from each node, its parent and grandparent. Knowledge-engineered approaches have been employed for problems for which there were insufficient data to train models, for certain graph and tree manipulation operations, and for straightforward operations such as performing contractions.

The next major stage fleshes out the logical form tree. Each node is labeled (NP, VP, SUBCL etc). Function words are inserted, including placeholders for determiners, e.g., *DefDet*, which serves as a placeholder for a definite determiner and *RelPro*, which serves as a placeholder for a relative pronoun (the exact inflected forms cannot be determined until subsequent processing has resolved intra-constituent ordering and morphological case), auxiliaries, the infinitival marker *zu*, the subordinating conjunctions *dass* and *ob*, expletive subject *es*, reflexive pronouns, the adverbial *wie*, the negator *nicht*, and the passive markers *von* and *durch*. A simple function produces the contracted prepositional proforms such as *dadurch* and *damit*. Logical subjects and objects are assigned a probability

[2] Work in progress.

of being realized in the surface string. For example, logical subjects of infinitival clauses do not usually receive overt expression. Finally, morphological case features and verb position features (verb second, verb final, verb initial) are assigned.

The third major stage deletes logical subjects and objects whose probability of being realized is below a given threshold. The degraphed logical form is transformed deterministically into a representation that more closely resembles a syntactic constituent representation, albeit with linear order still unspecified. Non-terminal nodes are given the syntactic labels that were chosen previously. A function splits separable prefixes from verbal stems, based on verb-position features assigned in the previous stage, and based on lexical information. A simple tree transformation reverses the headedness of constituents in which the syntactic head is not the semantic head, e.g., in cases involving quantification such as *viele der Leute* "many of the people".

Processing proceeds to global movement operations, i.e., movement that reattaches nodes, as distinct from ordering within a constituent. Mandatory movement, whether raising, Wh-movement, or the movement of relative pronouns/relative expressions, are handled by three simple functions—these phenomena, which have received considerable attention in the theoretical literature, have extremely low text frequency in the technical corpus that we use, with the result that there were insufficient examples for training decision tree classifiers. Extraposition of relative, infinitival and complement clauses is based on a decision tree classifier which decides for each instance of such a clause whether it should move up one step and attach to the parent of its parent (Gamon et al., 2002b). Once reattached there, the next "hop" is evaluated, until a position is found where the probability of further movement is less than the probability of staying put (i.e., it falls below 0.5).

Now that the basic constituency has been resolved, two functions set morphological properties of verbs, copying tense information to the finite verb (which might be an auxiliary that was inserted during flesh-out), and copying participial information onto the non-finite verb. The verb is made to agree with the subject, which is identified as the first nominative NP in the domain of the finite verb.

Next, linear order within each constituent is determined by consulting a generative statistical language model (see Ringger et al., in preparation (A)). For a given constituent, the ordering stage employs a beam search to select the optimal ordering of modifiers relative to one another and to the head, in the context of the syntactic category of the parent constituent and the head part-of-speech. In addition to leveraging the constituency, the order model is sensitive to the semantic relation between each modifier and the head as well as the syntactic category of the modifier.

A series of operations cleans up the tree. Decision tree classifiers select the appropriate forms to replace the placeholders for determiners and relative pronouns. This is a non-trivial task in German—the training data contain 55 different lexical realizations of determiners and 23 different lexical realizations of relative pronouns. Reflexive pronouns, which also received an abstract form during insertion in flesh-out, are converted into their surface form by a simple function. Finally, we must perform conjunction reduction, reducing sentences such as the grammatical but dysfluent *Hans hat die Kartoffeln gekocht, und Hans hat die Kartoffeln gegessen* "Hans has cooked the potatoes, and Hans has eaten the potatoes" to the more fluent *Hans hat die Kartoffeln gekocht und gegessen* "Hans has cooked and eaten the potatoes". A decision tree classifier assigns a probability to each NP under coordination that it will be overtly realized. A simple function then selects only the most likely NP to be realized. A final housekeeping function removes duplicated prepositions and auxiliaries that remain as artifacts of earlier degraphing operations.

Although there are extensive prescriptive descriptions of German punctuation conventions (e.g., Duden, 2000), we have implemented two decision tree classifiers. One classifier decides whether a punctuation symbol should be inserted after a given constituent. The other classifier decides whether a symbol should be inserted before a given constituent. The most probable symbol (including the NULL symbol) is inserted between each terminal node. Finally, a knowledge-engineered module performs morphological inflection. The output sentence is then read off from the terminal nodes of the tree.

5 Evaluation

Table 1 presents the accuracy of the various decision tree classifiers. Extended discussion of these and other classifiers is given in Gamon et al. (in preparation), including precision/recall figures for each value of the target feature. The baseline is calculated by predicting the most frequent value in the held-out test data. This proved easier to calculate from the intermediate WinMine data files, and is a slightly higher bar than predicting the most frequent value from the training data. All decision tree classifiers outperform the baseline. The accuracy for the extraposition model is based on a single “hop”, i.e., for a single decision concerning whether an extraposable clause ought to ascend one level in the syntax tree. For a more extensive evaluation that considers the final attachment site after a clause has undergone multiple hops, see Gamon et al. (2002b).

We evaluate the overall system by parsing a blind test set of 564 German sentences to produce logical forms and then applying Amalgam to generate output strings from those logical forms. For this sample, 71.1% of the words are correctly inflected and occur in the correct position in the sentence. We also compute the word-level string edit distance of the generated output from the original reference string: the number of errors (insertions, deletions, and substitutions) is 44.7% of the number of words in the reference string.

Classifier	**Values predicted**	**Accuracy**	**Baseline**
Syntactic label	Determiner phr., complement cl., VP, quantifier phr., adverbial NP, imperative main cl., adverb phr., label, appositive NP, question main cl., auxiliary phr., nominal relative, adjective phr., abbreviated cl., relative cl., NP, possessor, present participial cl., comment, infinitival cl., PP, finite subordinate cl., declarative main cl., past participial cl.	0.9823	0.3480
Placeholder for determiner	NULL, Wh, proximal demonstrative, definite, indefinite	0.9763	0.5769
Auxiliary	NULL, sein_werden, sein, haben, werden	0.9986	0.8136
Passive marker	NULL, von, durch	0.9915	0.9704
Insert *zu*	Yes, No	0.9977	0.9566
Insert negator	Yes, no	0.9094	0.8079
Subordinating conj	NULL, dass, ob	0.9547	0.5455
Insert expletive *es*	Yes, No	0.9965	0.9949
Realization of NP	Yes, No	0.8859	0.6819
Morphological case	Dat, Acc, Gen, Nom	0.9602	0.4626
Extraposition	Yes, No	0.8820	0.6711
Determiner form	Das, dasselbe, dem, demselben, den, denjenigen, denselben, der, derjenigen, derselbe, derselben, des, desselben, die, diejenige, diejenigen, dies, diese, dieselbe, dieselben, diesem, diesen, dieser, dieses, ein, ein_und_demselben, ein_und_derselbe, ein_und_derselben, eine, einem, einen, einer, eines, ihr, ihre, ihrem, ihren, ihrer, ihres, mein, meine, meinem, meiner, meines, sein, seine, seinem, seinen, seiner, seines, welche, welchem, welchen, welcher, welches	0.9077	0.2165
Relative pron. form	das, das/der, dem, den, denen, der, deren, derer, dessen, die, die/der, warum, was, welche, welcher, welches, wo, wobei, wodurch, woher, womit, woraus, wozu	0.8779	0.5359
Conjunction reduction	Spell out: first, last or medial instance	0.9710	0.8501
Preceding punctuation	NULL, comma, dash, semi-colon, colon, others	0.9865	0.8961
Following punctuation	NULL, comma, dash, semi-colon, colon	0.9848	0.9498

Table 1 Accuracy of decision classifiers

String edit distance is a harsh measure of sentence realization accuracy. Because string edit distance does not consider movement as an edit operator, movements appear as both deletions and insertions, yielding a double penalty. Furthermore, as observed earlier, some edits have a greater impact on the intelligibility of the output than others, especially the position of the German verb. Work in progress on a tree edit distance metric addresses both of these issues (Ringger et al., in preparation (B)). Closely related work includes that of Bangalore, Rambow and Whittaker (2000).

It is possible that generated sentences might differ from the reference sentences and yet still prove satisfactory. We therefore had five independent human evaluators assess the quality of the output for a random sample of 564 sentences.[3] These sentences had been analyzed to yield logical forms from which Amalgam generated output sentences. The evaluators assigned an integer score to each sentence, comparing it to the reference sentence using the scoring system given in Table 2.[4]

The average score was 2.96 with a standard deviation of 0.81. The mode was 4, occurring 104 times, i.e. 104/564 sentences, or 18.4% received the maximum score. In 63 of these cases, the score of 4 had been automatically assigned because the output sentence was identical to the reference sentence. In the other 41 cases, all five human evaluators had assigned a score of 4, i.e., the output differed from the reference sentence, but was still "Ideal".

1. "Unacceptable". Absolutely not comprehensible and/or little or no information transferred accurately.
2. "Possibly Acceptable". Possibly comprehensible (given enough context and/or time to work it out); some information transferred accurately.
3. "Acceptable". Not perfect (stylistically or grammatically odd), but definitely comprehensible, AND with accurate transfer of all important information.
4. "Ideal". Not necessarily a perfect translation, but grammatically correct, and with all information accurately transferred.

Table 2 Evaluation guidelines

6 Conclusion

Some elements of the approach to sentence realization presented here are peculiar to our logical form representations, e.g., the fact that negation is represented as a feature on a node rather than as an operator with scope over a sub-graph, or the procedural operations necessary to prune artifacts introduced by the degraphing process. The more linguistic aspects of the procedure deal with issues that would be problems for any approach that proceeds from a representation of propositional content to a surface string, such as extraposition and conjunction reduction. As with more traditional knowledge-engineered approaches to sentence realization (e.g., Aikawa et al. 2001), we have attempted to model such linguistic phenomena in terms of syntactic constituency. Unlike these traditional approaches we have implemented the solutions to these problems in machine-learned components.

Direct comparison to the Nitrogen system (Langkilde and Knight, 1998a, 1998b) is instructive at this point, since in some aspects it is the system that is most philosophically similar to Amalgam. The knowledge-engineered component in Nitrogen that proposes candidate sentences inserts function words and inflectional variants of existing words. Although this makes the generation process robust to underspecified input, the search space rapidly explodes. If the addition of inflectional variants for English, a language with a relatively impoverished repertoire of inflectional morphemes, can lead to hundreds of thousands of

[3] We extract a random sample of generated sentences. We take the first *n* sentences in the sample necessary to ensure 500 sentences that differ from the reference sentence.

[4] These guidelines were originally intended for assessing the quality of machine translation, measuring fluency and transfer of semantic content from the source language. Evaluating the sentence realization component is conceptually a case of German-to-German translation.

candidate output sentences in the examples that Langkilde and Knight give, it is doubtful whether it would be practical for a language such as German with a rich inflectional morphology. We believe that a linguistically informed approach allows for a more direct model of the underlying phenomena, which constrains the search space and reduces the burden on the generative language model by presenting it with fewer spurious candidate sentences.

Although our exposition has focused on the preferred value (the mode) predicted by the models, decision trees built by WinMine predict a probability distribution over all possible target values. For a system such as Amalgam, built as a pipeline of stages, this point is critical, since finding the best final hypothesis will require the consideration of multiple hypotheses and the concomitant combination of probabilities assigned by the various models in the pipeline to all possible target values. In its current implementation, Amalgam follows a greedy search, but in future research we intend to experiment with a beam search that entertains multiple intermediate hypotheses.

Acknowledgements

Our thanks go to Max Chickering for assistance with the WinMine decision tree tools, to Zhu Zhang who made significant contributions to the development of the extraposition and punctuation models and to Karin Berghöfer and the five anonymous evaluators from the Butler Hill Group for assistance with evaluation.

References

T. Aikawa, M. Melero, L. Schwartz, and A. Wu. 2001. Multilingual Sentence Generation. In Proceedings of 8th European Workshop on Natural Language Generation. Toulouse, France. 57-63.

S. Bangalore and O. Rambow. 2000a. Exploiting a probabilistic hierarchical model for generation. In Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000). Saarbrücken, Germany. 42-48.

S. Bangalore and O. Rambow. 2000b. Corpus-based lexical choice in natural language generation. In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL 2000). Hong Kong, PRC. 464-471.

S. Bangalore, O. Rambow, and S. Whittaker. 2000. Evaluation metrics for generation. International Conference on Natural Language Generation (INLG 2000). Mitzpe Ramon, Israel. 1-13.

D. Max. Chickering. nd. WinMine Toolkit Home Page. http://research.microsoft.com/~dmax/WinMine/Tooldoc.htm

S. Corston-Oliver. 2000. Using Decision Trees to Select the Grammatical Relation of a Noun Phrase. In Proceedings of the 1st SIGDial workshop on discourse and dialogue. Hong Kong, PRC. 66-73.

P. Duboue and K. McKeown. 2001. Empirically estimating order constraints for content planning in generation. In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001). Toulouse, France. 172-179.

Duden. 2000. Die deutsche Rechtschreibung. Duden-Verlag: Mannheim, Leipzig, Wien, Zürich.

M. Gamon, E. Ringger and S. Corston-Oliver. 2002a. Amalgam: A Machine-learned Generation Module. Microsoft Research Tech Report. To appear.

M. Gamon, S. Corston-Oliver, E. Ringger, Z. Zhang, B. Moore. 2002b. Extraposition of relative clauses: A case study in German sentence realization. To appear in Proceedings of COLING 2002, Taipei, Republic of China.

G. Heidorn. 2000. Intelligent Writing Assistance. In R. Dale, H. Moisl, and H. Somers (eds.), A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text. Marcel Dekker, New York. 181-207.

I. Langkilde and K. Knight. 1998a. The practical value of n-grams in generation. In Proceedings of the 9th International Workshop on Natural Language Generation, Niagara-on-the-Lake, Canada. 248-255.

I. Langkilde and K. Knight. 1998b. 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 (COLING-ACL 1998). Montréal, Québec, Canada. 704-710.

J. Oberlander and C. Brew. 2000. Stochastic text generation. To appear in Philosophical Transactions of the Royal Society of London, Series A, volume 358.

M. Poesio, R. Henschel, J. Hitzeman and R. Kibble. 1999. Statistical NP generation: A first report. In Proceedings of the ESSLLI Workshop on NP Generation. Utrecht, Netherlands.

A. Ratnaparkhi. 2000. Trainable methods for surface natural language generation. In Proceedings of the 6th Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of the Association of Computational Linguistics (ANLP-NAACL 2000). Seattle, Washington, USA. 194-201.

E. Ringger, B. Moore, M. Gamon, S. Corston-Oliver. In preparation (A). A linguistically-informed generative language model for intra-constituent ordering during sentence realization.

E. Ringger, R. Moore, M. Gamon, S. Corston-Oliver. In preparation (B). A tree edit distance metric for evaluation of natural language generation.

H. Uszkoreit, T. Brants, D. Duchier, B. Krenn, L. Konieczny, S. Oepen and W. Skut. 1998. Aspekte der Relativsatzextraposition im Deutschen. Claus-Report Nr.99, Sonderforschungsbereich 378, Universität des Saarlandes, Saarbrücken, Germany.

M. Walker, O. Rambow, and M. Rogati. 2001. SPoT: A trainable sentence planner. In Proceedings of the North American Meeting of the Association for Computational Linguistics.

I. Zukerman, R. McConachy and K. Korb. 1998. Bayesian reasoning in an abductive mechanism for argument generation and analysis. AAAI98 Proceedings—the Fifteenth National Conference on Artificial Intelligence, pp. 833-838, Madison, Wisconsin.