The Types and Distributions of Errors in a Wide Coverage Surface Realizer Evaluation

Charles B. Callaway
HCRC, University of Edinburgh
2 Buccleuch Place
Edinburgh, UK EH8 9LW
ccallawa@inf.ed.ac.uk

Abstract

Recent empirical experiments on surface realizers have shown that grammars for generation can be effectively evaluated using large corpora. Evaluation metrics are usually reported as single averages across all possible types of errors and syntactic forms. But the causes of these errors are diverse, and the extent to which the accuracy of generation over individual syntactic phenomena is unknown.

This article explores the types of errors, both computational and linguistic, inherent in the evaluation of a surface realizer when using large corpora. We analyze data from an earlier wide coverage experiment on the FUF/SURGE surface realizer with the Penn TreeBank in order to empirically classify the sources of errors and describe their frequency and distribution. This both provides a baseline for future evaluations and allows designers of NLG applications needing off-the-shelf surface realizers to choose on a quantitative basis.

1 Introduction

Surface realization is the process of converting the semantic and syntactic representation of a sentence or series of sentences into a surface form for a particular language. Most reusable deep surface realizers [Elhadad, 1991; Bateman, 1995; Lavoie and Rambow, 1997; White and Caldwell, 1998] have been symbolic, hand-written grammar-based systems, often based on syntactic linguistic theories such as Halliday's [Halliday, 1976] systemic functional theory (FUF/SURGE and KPML) or Mel'cuk's [Mel'cuk, 1988] Meaning-Text Theory (REALPRO).

However, corpus-based components, and in particular statistical surface realizers [Langkilde and Knight, 1998; Bangalore and Rambow, 2000; Ratnaparkhi, 2000; Langkilde-Geary, 2002] have focused attention on a number of problems facing symbolic NLG systems that until now have been generally considered future work: large-scale, data-robust and language- and domain-independent generation. In each case, empirical evaluation plays a fundamental role in determining performance at the task of surface realization and setting baselines for future performance evaluation.

For instance, the HALOGEN statistical realizer [Langkilde-Geary, 2002] underwent the most comprehensive evaluation of any surface realizer, which was conducted by measuring sentences extracted from the Penn TreeBank [Marcus *et al.*, 1993], converting them into its input formalism, and then producing output strings. Using automatic metrics from machine translation then quickly produces figures for global characteristics of traits such as accuracy.

In a recent experiment, we compared the performance of HALOGEN relative to the grammar-based FUF/SURGE surface realizer on the identical corpus and with a similar methodology [Callaway, 2003; 2004]. Although FUF/SURGE scored higher than the HALOGEN realizer, we were interested in absolute as well as relative performance: *e.g.*, what particular grammatical rules are not well-covered by SURGE's grammar?

While the above methodology gives an average figure for what coverage and accuracy are on a corpus like the Penn TreeBank, it describes a very wide array of errors as simple numerical averages. Thus it is impossible to know without working experience which types of realizer errors HALOGEN is likely to make. From the perspective of syntactic analysis, statistical realizers behave as black boxes and thus there is little or no attempt made to look at incorrect sentences to trace back exactly what caused a particular error. Instead, either the language model is adjusted or a new corpus is obtained. However, the grammars in symbolic realizers can also be evaluated as glass boxes, allowing the improvement of an incorrect or missing grammatical rule to also improve the generation of all other instances of the same grammatical rule in the corpus.

Those who are looking to use a surface realizer typically fall into three cases: (1) those employing one for the first time, often looking to use a well-known system, (2) those seeking to use a different surface realizer because their current realizer has undesirable operational parameters such as slowness or lack of multilingual support, and (3) those seeking to change realizers due to lack of grammatical coverage for their domain.

Those falling in this latter category are not interested in a general quantitative measure for coverage, but rather in finding out, without expending too much effort, whether a given surface realizer will generate the types of sentences they need. We were thus interested in a range of questions about the evaluation itself as well as the results of the evaluation, which

Realizer	Sentences	Coverage	Exact Matches	SS Accuracy	BLEU Accuracy
SURGE 2.2	1192	49.5%	368 (30.9%)	0.8206	0.7350
SURGE 2.3	2372	98.5%	1644 (69.3%)	0.9605	0.9321
HALOGEN	1968	82.8%	1132 (57.5%)	0.9450	0.9240

Table 1: Comparing two SURGE versions with HALOGEN on 2416 sentences from Section 23 of the Penn TreeBank.

would allow a number of questions to be answered:

- What types of problems might be encountered during the experiment itself, and what are their sources?
- What kinds of constructions are difficult to handle in general, such as adverb positioning, verb argument ordering, or very long noun phrases?
- What coverage does FUF/SURGE have for specific types of infrequent syntactic phenomena, like indirect questions or topicalizations?
- What is a reasonable distribution of these errors?
- What data is necessary to allow one to predict if a particular surface realizer will match the expectations of a new application?

We look at these questions from two perspectives: (1) establishing baselines to inform future, more detailed evaluations, and (2) providing information about the current state of the FUF/SURGE grammar and morphology for those who may wish to use it as a surface realizer in a new project and need to know whether it will support the types of syntactic constructions needed in their domain.

Indeed, for some domains and genres, particular phenomena like direct questions may be more important than overall coverage, and thus when creating an application for those domains, a surface realizer should be chosen with these data for these phenomena in mind. For instance, domains involving dialogue must frequently generate sentence fragments rather than complete sentences, although this latter type is the subject of the most scrutiny in surface realizer research.

To provide a foundation for answering these questions, we performed a manual analysis on a set of specific errors in sentences generated from the Penn TreeBank by the FUF/SURGE surface realizer. To do this, we generated 4,240 sentences, selecting those which did not match the target sentence and had a high probability of being incorrect due to problems with wrong or missing syntactic rules in the grammar. Each of the resulting sentences was analyzed to determine the source of the error, and in the case of poor grammatical coverage, the missing or incorrect syntactic construction that was at fault.

We conclude that grammatical errors are actually a small part of the errors found, and in particular, four types of bad grammatical rules were responsible for almost two thirds of accuracy errors. But first we describe in more detail how sentences were generated from the corpus, how they are measured for accuracy, and what high-level types of errors prevent sentences from being perfectly realized.

2 Methodology

Undertaking a large-scale evaluation for a symbolic surface realizer requires a large corpus of sentence plans. Since text planners cannot generate either the requisite syntactic variation or quantity of text, [Langkilde-Geary, 2002] developed an evaluation strategy for HALOGEN employing a substitute: sentence parses from the Penn TreeBank [Marcus *et al.*, 1993], a corpus that includes texts from newspapers such as the Wall Street Journal, and which have been hand-annotated for syntax by linguists.

However, surface realizers typically have idiosyncratic input representations, and none use the Penn TreeBank parse representation. Thus a transformer is needed to convert the TreeBank notation into the language accepted by the surface realizer. As we were interested in comparing the coverage and accuracy of FUF/SURGE with Langkilde's HALOGEN system, we implemented a similar transformer [Callaway, 2003] to convert Penn TreeBank notation into the representation used by FUF/SURGE .

As with the HALOGEN evaluation, we used Simple String Accuracy [Doddington, 2002] and BLEU [Papineni *et al.*, 2001] to determine the average accuracy for FUF/SURGE . To obtain a meaningful comparison, we utilized the same approach as HALOGEN, treating Section 23 of the TreeBank as an unseen test set. A recent evaluation showed that the combination of the transformer and an augmented version of FUF/SURGE had higher coverage and accuracy (Table 1) compared to both HALOGEN and version 2.2 of FUF/SURGE .

The difference between the two versions of FUF/SURGE was especially striking, with the augmented version almost doubling coverage and more than doubling exact match accuracy. One example of these differences is the addition of grammatical rules for direct and indirect written dialogue, which comprise approximately 15% of Penn TreeBank sentences, and which is vital for domains such as written fiction.

However, this evaluation method does not allow for seamless comparisons. Inserting a transformation component between the corpus and realizer means that not only is the surface realizer being evaluated, but also the accompanying transformation component itself. We were thus interested in determining what proportion of reported errors could be attributed to the surface realizer as opposed to the transformer or to the corpus. To do this, as well as to assist application designers in better interpreting the results of these formal evaluations, we needed to identify more precisely what types of failures are involved.

We thus undertook a manual analysis of errors in Sections 20–22 by hand, individually examining 629 erroneous sentences to determine the reason for their failure. Although more than 629 out of the 5,383 sentences in these development sections produced accuracy errors, we eliminated approximately 600 others from consideration:

- *Simple string accuracy less than 10 characters*: Sentences with very small error rates are almost always incorrect due to errors in morphology, punctuation, and capitalization. For instance, a single incorrect placement of quotation marks has a penalty of 4. Thus it made little sense to manually examine them all in the off chance a handful had true syntactic errors.
- *Sentences of less than 10 or more than 35 words*: Sentences with 9 or fewer words were extremely unlikely to contain complex syntactic constructs, and collectively had an accuracy of over 99.1%. Sentences larger than 35 words with errors typically had more than one major grammatical error, making it very difficult to determine a single "exact" cause. 96% of sentences within the range had a single grammatical cause, and the remaining 4% had only 2 syntactic errors.

Each error instance in the resulting 629 sentences was classified twice: first to find the source of the error (corpus, transformer, or grammar), and then in the case of grammatical errors, to note the syntactic rule that caused the error. These two classifications are discussed in the following two sections. We were not realistically able to perform a similar comparison for coverage errors, because out of the 4,240 sentences satisfying the second criterion of sentence length, only 17 of them did not generate some string.

3 Types of Methodological Errors

Errors in the surface realizer evaluation, which can manifest themselves either as empty sentences or as generated sentences which do not exactly match the target string, can arise from the corpus itself, the transformation component, or the surface realizer, which consists of the grammar, linearization rules, and the morphology component.

The corpus itself can be a source of errors due to two main reasons: (1) the corpus annotators have incorrectly analyzed the syntactic structure, for instance, attaching prepositions to the wrong head, or including grammatically impossible rules, such as `NP → VB CC VB`, or (2) the parts of speech were mistagged by the automatic POS tagger and were not corrected during the supervision process, as in `(NP (NN petition) (VBZ drives))`.

Unfortunately, the corpus cannot easily be cleaned up to remove these errors, as this significantly complicates the comparison of results across corpus versions. We must thus subtract the proportion of corpus errors from the results, creating a "topline" which defines a maximum performance measure for the realizer. Manually analyzing incorrect sentences produced from the corpus allows this topline to be determined with reasonable accuracy.

In addition to errors in the corpus, other types of errors originate in the transformation component, as it attempts to match TreeBank annotations with rules that produce the requisite input notation for the surface realization. While such transformers are highly idiosyncratic due to differing input and output notations, the following categories are abstract, and thus likely to apply to many different transformers.

- *Missing Tag*: While there is a standardized set of tags, the semantic subtags and coreference identifiers can combine to create unpredictable tags, such as `PP-LOC-PRD-TPC-3` or `PP-EXT=2`.
- *Missing Rule*: Often each of the individual tags are recognized, but no rule exists to be selected for a given ordered combination of tags, like `ADVP-MNR → RB COMMA RB RB`.
- *Incorrect Rule*: The transformation component may select the wrong rule to apply, or the rule itself may be written incorrectly or may not have been written with all possible combinations of tag sequences in mind.
- *Ordering*: Some phrasal elements such as adverbial clauses can be placed in five or even six different positions in the matrix clause. Choosing the wrong position will result in errors reported by the automatic accuracy metrics, as discussed in [Callaway, 2003]. An important note is that the order can be incorrect but still make sense semantically.

Finally, even given a correct input representation, a surface realizer can also produce errors during the realization process. Of the four main surface realizer functions below, only syntactic rules provide a significant source of accuracy errors from the point of view of averaged metrics:

- *Syntactic Rules*: The grammar may be missing a particular syntactic rule or set of features, or may have been encoded incorrectly. For instance, the stock version of FUF/SURGE did not have a rule allowing noun phrases to terminate in an adverb like "ago" as in "five years ago", which occurs frequently in the Penn TreeBank, causing the word to be missing from the generated sentence.
- *Morphology*: While morphological errors occasionally appear, they are usually very small and do not contribute much to the overall accuracy score. The most common problems are irregular verbs, foreign plural nouns, and the plurals of acronyms, as well as the marking of acronyms and y/u initial letters with indefinite a/an.
- *Punctuation*: While most errors involving punctuation marks also contribute very little statistically to the overall score of a sentence (*e.g.*, a missing comma), the TreeBank also contains combinations of punctuation like long dashes followed by quotation marks. Additionally, incorrect generation of mixed quotations can lead to repeated penalties when incorrectly determining the boundaries of the quoted speech, and large penalties if the multiple forms of punctuation occur at the same boundary [Callaway, 2003].
- *Linear Precedence*: In our analysis of realizer errors, no examples of obligatory precedence violations were found (as opposed to "Ordering" problems described above.)

4 Types of Syntactic Errors

While general types of errors in the evaluation process are helpful for improving future evaluations, a more pressing question for those wishing to use an off-the-shelf surface realizer is how well it will work in their own application domain. The coverage and accuracy metrics used by Langkilde are very broad measures which say nothing about the effectiveness of a surface realizer when generating individual syntactic constructions. The advantage of these metrics are that they are easy to compute over the entire corpus, but lose this capability when the same question is asked about particular subsets of a general corpus, such as all sentences containing an indirect question.

When performing an analysis of the types of syntactic errors produced by FUF/SURGE when given correct inputs, we found nine syntactic constructions that resulted in at least two or more sentences being generated incorrectly. The analysis allows us to conclude that FUF/SURGE is either not reliably capable or else incapable of correctly producing the following syntactic constructions (a manual analysis of all 5,383 sentences to find all correct instances of these constructions is impractical, although we were able to automate some corpus searches based on particular semantic tags):

- *Inversion*: Pragmatic inversions of auxiliaries in embedded questions [Green, 2001] or in any other construction besides questions, negations, and quoted speech. Thus a TreeBank sentence like "This is the best time to buy, *as was the case* two years ago." cannot be generated.
- *Missing verb tense*: While FUF/SURGE has 36 predefined verb tenses, the corpus contained several instances of another tense: "…which fellow officers remember as *having been* $300."
- *Mixed conjunctions*: Often in the Penn TreeBank the UCP tag (unlike coordinated phrase) marks conjunctions where the constituents are not all of the same grammatical category, but in compound verb phrases, they are often marked as simple conjunctions of mixed types. But FUF/SURGE requires verb phrases in conjunctions to be compatible on certain clausal features with all constituents, which is violated in the following example: [VP $\rightarrow$ VP CC VP COMMA SBAR-ADV] "Instead, they bought on weakness and sold into the strength, *which kept the market orderly*."
- *Mixed type NP modifiers*: FUF/SURGE's NP system assumes that cardinal numbers will precede adjective modifiers, which will precede nominal modifiers, although the newspaper texts in the Penn TreeBank have more complex NPs than were considered during the design of the NP system: "a $100 million Oregon general obligation veterans' tax note issue".
- *Direct questions*: Direct questions are not very common in newspaper text, in fact there are only 61 of them out of the entire 5,383 sentences of Sections 20–22. More complex questions involving negations and modal auxiliaries are not handled well, for example "Couldn't we save $20 billion by shifting it to the reserves?" though simpler questions are generated correctly.
- *Indirect questions*: The Penn TreeBank contains a roughly equivalent number of instances of indirect questions as direct, such as "It's a question of how much credibility you gain." and again the reliability of generating this construction depends on the complexity of the verbal clause and the question phrase.
- *Mixed level quotations*: One of the most difficult syntactic phenomena to reproduce is the introduction of symmetric punctuation that cuts across categorial boundaries [Doran, 1998]. For instance, in the following sentence, the first pair of quote marks are at the beginning of an adverbial phrase, and the second pair are in the middle, separating two of its constituents: *…the U.S. would stand by its security commitments "as long as there is a threat" from Communist North Korea.*
- *Complex relative pronouns*: While simple relatives are almost always handled correctly except in certain conjunctions, complex relatives like partitive relatives ("all of which", "some of which"), relatives of indirect objects or peripheral verbal arguments like locatives ("to whom", "in which"), complex possessives ("whose $275-a-share offer") and raised NP relatives ("…the swap, details of which…") were not considered when FUF/SURGE was designed.
- *Topicalization*: The clausal system of FUF/SURGE is based on functional grammar [Halliday, 1976], and so does not expressly consider syntactic phenomena such as left dislocation or preposing of prepositional phrases. Thus sentences like "Among those sighing with relief was John H. Gutfreund" may generate correctly depending on their clausal thematic type, like `material` or `equative`.

While we present the results of a manual analysis of the data in the next section, it is important to remember that the large majority of syntactic constructions, punctuation and morphology worked flawlessly in the evaluation of FUF/SURGE as described in [Callaway, 2004]. As described earlier, almost 7 out of every 10 sentences in the unseen test set were exact matches, including punctuation and capitalization. Additionally, most errors that did occur were in the transformation component rather than the surface realizer, as we will describe shortly. Finally, some well-studied but rare syntactic constructions did not occur in the sections of the Penn TreeBank that we examined, such as left dislocation and negative NP preposing.

5 Data Analysis

As mentioned previously, we undertook a manual analysis of Sections 20–22 of the Penn TreeBank by hand to determine specific reasons behind the failure of 629 sentences out of 4,240 that met the criteria of having between 15 and 44 words, and having a character error rate of more than 9 as determined by the SSA metric.

Table 2 presents the results for high-level error types as described in Section 3. It shows that the greatest proportion of errors is due to the transformation process: 390 sentences (62.0%) or 15,733 (63.4%) of the character-based accuracy error. This is expected given that the transformation component has been developed in a year or so, while FUF/SURGE has been in use for around 15 years. Each of the 166 sentences that were incorrect due to inaccurate transformer rules was verified by ensuring that the sentence would correctly generate with minor changes to the automatically produced

Error Type	# Occurrences		Total SSA Penalty		Avg. Penalty
Corpus Error	40	6.36%	1922	7.74%	48.05
Transformer Rule Error	166	26.39%	7201	29.01%	43.38
No Transformer Tag	12	1.91%	679	2.74%	56.58
No Transformer Rule	102	16.22%	4320	17.40%	42.35
Ordering (Good)	55	8.74%	2137	8.61%	38.85
Ordering (Bad)	55	8.74%	1396	5.62%	25.38
Punctuation/Morphology	14	2.23%	389	1.57%	27.79
Syntax	185	29.41%	6777	27.30%	33.70
Total	629	100.0%	24821	100.0%	38.60

Table 2: Distribution of 629 high-level errors in the 4,240 tested sentences from Sections 20–22.

functional description. The error rate of the Penn TreeBank annotation is a reasonably well-known quantity, and there is a specialized literature describing automatic correction methods (*e.g.*, [Dickinson and Meurers, 2003]).

One surprise though is that while the number of errors due to the ordering of floating constituents is the same, the error in accuracy is skewed to semantically acceptable interpretations. And while the distribution of the order seems like random chance, it should be remembered that there can potentially be up to 10 acceptable placements when there are multiple floating constituents. Additionally, unrecognized annotation tags seem to invoke the heaviest average penalty for any error type, but have the lowest rate of occurrence.

Some advice then for future evaluations of this type would be to systematically ensure that all tags are normalized in the corpus before writing transformation rules. Missing transformation rules were always single-case errors, and a large amount of effort would need to be expended to account for them, following the well-known 80/20 rule. The data in Table 3 then allows other surface realizer researchers to prioritize their time when developing their own evaluations.

Finally, slightly over a quarter of the reduction in accuracy is due to syntactic phenomena that are not handled correctly by the surface realizer. Given that this error category is most of interest in determining which surface realizer has the necessary coverage for a particular domain, we investigated further the interactions between error rates and individual syntactic phenomena.

Table 3 presents the number of occurrences of errors for each of the syntactic phenomena presented in the previous section. We can see that topicalizations, direct questions and inversions were on average most likely to produce the largest error per instance, at 73.18, 56.00 and 50.08 edit distances each. The most frequent error types were mixed NP modifiers, but such constructions were small enough (often involving only two words in switched order) that they had the second lowest SSA penalty.

Knowing the ratios of errors allows those weighing different surface realizers for a new project to select based on a number of criteria. For instance, in some domains, it may be undesirable to have the reader see a large number of surface language errors where the extent of each error is unimportant, whereas in other situations, large mistakes that completely obscure the intent of the sentence are more of a problem.

While Table 3 tells us which syntactic type is most likely to produce the largest accuracy penalty, it does not tell us which syntactic types are most frequent in the corpus, since this would require also counting all correct instances, which would be very prohibitive to do manually and inaccurate to do automatically. Knowing this quantity would be of greatest help to an NLG application designer wanting to compare surface realizers, but is difficult to do in practice.

We thus decided to look at correct instances of a small number of rare phenomena which can easily be found by searching for tags in the TreeBank. For instance, it-clefts are marked with the annotation `S-CLF`, of which there are 4 in the 5,383 sentences in Sections 20–22. However, by searching through the text representations with the regular expression `it is * that` and `it was * that`, we found an additional 2 it-clefts that were incorrectly marked (although all 6 examples were exact matches when generated by the surface realizer). By a similar process, we discovered 7 marked and 1 unmarked wh-clefts, which also were exact matches. A further investigation for topicalized sentences uncovered 6 instances that were correctly generated versus the 11 incorrectly generated.

The number of errors in the Penn TreeBank annotations on these rare constructions should give pause to those who want to create statistical selection algorithms from such data, given that the signal-to-noise ratio may be very high. Additionally, all of the data presented above reflects only this corpus; spoken dialogue corpora may vary significantly in frequencies of topicalization and left dislocation, for example.

6 Conclusions

Recent empirical experiments on surface realizers have shown that grammars for generation can be effectively evaluated using large corpora. We have helped clarify to what extent errors in accuracy may be due to the corpora itself and in the transformation process necessary to convert annotated sentences into the surface realizer's notation. Furthermore, we have performed a set of quantitative, manual analyses that have classified with increasing rigor the types of syntactic phenomena missing from the generation grammar of FUF/SURGE . The results demonstrate that FUF/SURGE is surprisingly robust with coverage lacking for a few important but somewhat infrequent syntactic phenomena. Finally, we

Error Type	# Occurrences		Total SSA Penalty		Avg. Penalty
Inversion	12	6.22%	601	8.99%	50.08
Missing Verb Tense	3	1.55%	47	0.70%	15.67
Mixed Conjunction	26	13.47%	1080	16.15%	41.54
Mixed NP Modifiers	64	33.16%	1197	17.90%	18.70
Question, Direct	10	5.18%	560	8.37%	56.00
Question, Indirect	9	4.66%	307	4.59%	34.11
Quotation, Unquoted	13	6.74%	542	8.11%	41.69
Quotation, Mixed	29	15.02%	1231	18.41%	42.45
Relative Clause	16	8.29%	317	4.74%	19.81
Topicalization	11	5.70%	805	12.04%	73.18
Total	193	100.0%	6687	100.0%	34.65

Table 3: Distribution of 193 major syntactic errors in the 4,240 tested sentences from Sections 20–22.

have established a topline and baseline performance measure for use in future comparisons between surface realizers.

References

[Bangalore and Rambow, 2000] S. Bangalore and O. Rambow. Exploiting a probabilistic hierarchical model for generation. In *COLING–2000: Proceedings of the 18th International Conference on Computational Linguistics*, Saarbruecken, Germany, 2000.

[Bateman, 1995] John A. Bateman. KPML: The KOMET-penman (multilingual) development environment. Technical Report Release 0.8, Institut für Integrierte Publikations- und Informationssysteme (IPSI), GMD, Darmstadt, 1995.

[Callaway, 2003] Charles B. Callaway. Evaluating coverage for large symbolic NLG grammars. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 811–817, Acapulco, Mexico, August 2003.

[Callaway, 2004] Charles Callaway. Wide coverage symbolic surface realization. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 125–128, Barcelona, Spain, July 2004.

[Dickinson and Meurers, 2003] M. Dickinson and D. Meurers. Detecting errors in part-of-speech annotation. In *Proceedings of the 10th Conference of the European Chapter of the ACL*, Budapest, Hungary, April 2003.

[Doddington, 2002] George Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the 2002 Conference on Human Language Technology*, San Diego, CA, March 2002.

[Doran, 1998] Christine Doran. *Incorporating Punctuation into the Sentence Grammar: A Lexicalized Tree Adjoining Grammar Perspective*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1998.

[Elhadad, 1991] Michael Elhadad. FUF: The universal unifier user manual version 5.0. Technical Report CUCS-038-91, Dept. of Computer Science, Columbia University, 1991.

[Green, 2001] Georgia Green. Pragmatic motivation and exploitation of syntactic rules. Technical report, University of Illinois, http://www.linguistics.uiuc.edu/g-green/441/prmoex/index.html, 2001.

[Halliday, 1976] Michael Halliday. *System and Function in Language*. Oxford University Press, Oxford, 1976.

[Langkilde and Knight, 1998] Irene Langkilde and Kevin Knight. Generation that exploits corpus-based statistical knowledge. In *COLING-ACL-98: Proceedings of the Joint 36th Meeting of the ACL andthe 17th International COLING*, pages 704–710, Montréal, Canada, August 1998.

[Langkilde-Geary, 2002] Irene Langkilde-Geary. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Second International Natural Language Generation Conference*, Harriman, NY, July 2002.

[Lavoie and Rambow, 1997] Benoit Lavoie and Owen Rambow. A fast and portable realizer for text generation systems. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, 1997.

[Marcus *et al.*, 1993] M. Marcus, B. Santorini, and M. Marcinkiewicz. Building a large annotated corpus of English: The PennTreeBank. *Computational Linguistics*, 26(2), 1993.

[Mel'cuk, 1988] Igor A. Mel'cuk. *Dependency Syntax: Theory and Practice*. SUNY Publications, 1988.

[Papineni *et al.*, 2001] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. BLEU: A method for automatic evaluation of MT. Technical Report RC22176, IBM Research, New York, September 2001.

[Ratnaparkhi, 2000] Adwait Ratnaparkhi. Trainable methods for surface natural language generation. In *Proceedings of the First North American Conference of the ACL*, Seattle, WA, May 2000.

[White and Caldwell, 1998] Michael White and Ted Caldwell. EXEMPLARS: A practical, extensible framework for dynamic text generation. In *Proceedings of the Ninth International Workshop on NLG*, pages 266–275, Niagara-on-the-Lake, Ontario, August 1998.