Evaluating Coverage for Large Symbolic NLG Grammars

Charles B. Callaway
ITC-irst Istituto per la Ricerca Scientifica e Tecnologica
via Sommarive, 18 - Povo
38050 Trento, Italy
callaway@itc.it

Abstract

After many successes, statistical approaches that have been popular in the parsing community are now making headway into Natural Language Generation (NLG). These systems are aimed mainly at surface realization, and promise the same advantages that make statistics valuable for parsing: robustness, wide coverage and domain independence. A recent experiment aimed to empirically verify the linguistic coverage for such a statistical surface realization component by generating transformed sentences from the Penn TreeBank corpus. This article presents the empirical results of a similar experiment to evaluate the coverage of a purely symbolic surface realizer. We present the problems facing a symbolic approach on the same task, describe the results of its evaluation, and contrast them with the results of the statistical method to help quantitatively determine the level of coverage currently obtained by NLG surface realizers.

1 Introduction

Like parsing, text generation offers enormous potential benefits for more natural interaction with computers. Examples of applications which could be greatly improved include automatic technical documentation, intelligent tutoring systems, and machine translation, among many others. Historically, natural language generation (NLG) has focused on the study of symbolic pipelined architectures which receive knowledge structures and goals from knowledge-based applications and which proceed to progressively add linguistic information.

In the last few years, the same paradigm shift which occurred in the parsing community, the use of statistical/empirical methods, has begun to influence the NLG community as well. As with parsing, statistical generation promises benefits such as robustness in the face of bad data, wider coverage, domain and language independence, and less need for costly resources such as grammars. But unlike parsing, which starts with a very flat representation (text) which is easily accessible in large quantities to both statistical and symbolic methods, the semantic input for NLG is typically associated with large knowledge-based systems. The types of corresponding corpora which would be necessary for using statistical processes, pairs of subgraphs of knowledge bases and their texts, do not currently exist in large quantities.

Because of this representation problem, most statistical systems have concentrated on replacing existing individual components in the standard NLG pipelined architecture [Reiter, 1994] without changing the remaining original symbolic modules. The most popular candidate has been the surface realization module [Elhadad, 1991; Bateman, 1995; Lavoie and Rambow, 1997; White and Caldwell, 1998], which is responsible for converting the syntactic representation of a sentence into the actual text seen by the user. Thus current statistical generators are still dependent on remaining architectural modules in a system to function and do not by themselves account for a large amount of linguistic phenomena: pronominalization, revision, definiteness, *etc.*

However, 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 standard, pipelined NLG that have until now been generally considered future work: large-scale, data-robust and language- and domain-independent generation. In addition, as Langkilde points out, empirical evaluation has not been standard practice in the NLG community, which has instead relied either on the software engineering practice of regression testing with a suite of examples or theoretical evaluations [Robin and McKeown, 1995].

This paper presents the analogue of this recent statistical experiment using a well-known off-the-shelf symbolic surface realizer, using an augmented generation grammar that includes support for dialogue and additional syntactic coverage. We first describe in the following section the representations and processes needed to understand its evaluation. We then detail our implemented system for converting sentences from a large corpus into a systemic functional notation, present an evaluation of that system and the grammar itself using Section 23 of the Penn TreeBank [Marcus *et al.*, 1993], and finally discuss the implications of that evaluation.

2 Sentence Representations

To undertake a large-scale evaluation of a symbolic surface realizer, we must first find a large quantity of sentence plans with which to produce text. However, most text planners cannot generate either the requisite syntactic variation or quantity of text, and we thus cannot turn to implemented gener-

```
(S (PP (IN Without)
       (NP (NNP GM)))
   (, ,)
   (NP-SBJ
       (NP (JJ overall) (NNS sales))
       (PP (IN for)
           (NP (DT the) (JJ other)
               (NNP U.S.) (NNS automakers))))
   (VP (VBD were)
       (ADJP-PRD (RB roughly) (JJ flat)
           (PP (IN with)
               (NP (CD 1989) (NNS results "))))))
```

```
((cat clause)
 (circum ((accompaniment ((cat pp) (position front) (accomp-polarity -)
                          (np ((cat proper) (lex "GM")))))))
 (process ((type ascriptive) (tense past)))
 (participants ((carrier ((cat common) (lex "sale") (number plural)
                          (describer ((cat adj) (lex "overall")))))
                          (qualifier ((cat pp) (prep ((lex "for")))
                                      (np ((cat common) (lex "automaker") (definite yes)
                                           (number plural) (status different)
                                           (classifier ((cat proper) (lex "U.S."))))))))))
                (attribute ((cat ap) (lex "flat") (modifier ((cat adv) (lex "roughly")))
                            (qualifier ((cat pp) (prep ((lex "with")))
                                        (np ((cat common) (lex "result") (number plural)
                                             (classifier ((cat date) (year 1989)))))))))))))
```

Figure 1: A Penn TreeBank Annotated Sentence and Corresponding FUF/SURGE Functional Description

ation systems as a source. To solve this problem, Langkilde trained a statistical algorithm [Langkilde-Geary, 2002] on a substitute set of sentence plans: the Penn TreeBank [Marcus *et al.*, 1993], a collection of sentences from newspapers such as the Wall Street Journal, which have been hand-annotated for syntax by linguists. An example sentence is shown on the left side of Figure 1. Hierarchical syntactic/semantic bracketing is provided along with the syntactic categories of lexemes and symbols in the newspaper texts.

Unfortunately, text planners currently in use do not generate representations of the form found in the Penn TreeBank, opting instead to use more fully-developed syntactic theories, such as HPSG [Pollard and Sag, 1994], from the linguistics community. Because annotated texts do not exist in this form, Langkilde created a pre-processing system to translate from the TreeBank annotation into the language accepted by the HALOGEN statistical surface realizer [Langkilde-Geary, 2002]. HALOGEN uses these translations to create a *forest lattice* whose paths from start to finish represent many possible versions of a single sentence. Separately, a larger corpus is processed to obtain bigram or trigram frequencies, which are then used to rank the possible sentence versions based on word adjacency. The highest ranked sentence is then presented as the final output of the system.

In contrast, most deep surface realizers are symbolic rather than statistical, and consist of components that check grammatical constraints, appropriately linearize constituents, and adjust for morphology and formatting. One such system in wide use, FUF/SURGE [Elhadad, 1991], combines ideas from systemic functional grammars and head-driven phrase structure grammars. An example of the FUF representation, known as a *functional description* is shown on the right side of Figure 1. SURGE is the largest generation grammar for English, and has the largest regression test suite available. But as Langkilde pointed out, 500 test examples are insufficient to empirically demonstrate the coverage of a grammar.

To arrive at a set of sentence plans which is representative of English, as well as to evaluate the coverage of the FUF/SURGE surface realizer in a way which can be directly compared to the HALOGEN evaluation, we likewise used the Penn TreeBank as a sentence source. Because our representations are also different, we (as Langkilde) needed a pre-processing system to convert from the TreeBank notation into the functional descriptions expected by the surface realizer.

Our pre-processor thus performs top-down structure traversal of a sentence annotated in Penn TreeBank format and

```
((S (-LRB- -LRB-)
    (NP-SBJ-1 (NP (NNP Sony) )
              (NP (PRP itself)))
    (VP (VBZ declines)
        (S (NP-SBJ (-NONE- *-1))
           (VP (TO to)
               (VP (VB comment)))
    (. .) (-RRB- -RRB-)))
```

```
(S (LRB LRB)
   (NP-SBJ-1 (NP (NNP "Sony"))
             (NP (PRP "itself")))
   (VP (VBZ "declines")
  (S (NP-SBJ (NONE STAR-1))
    (VP (TO "to")
        (VP (VB "comment")))))
  (RRB RRB)))
```

Figure 2: Penn TreeBank Notation and Normalized Form

builds the corresponding functional description. The pre-processor is organized as a context-sensitive, proceduralized rewriting grammar which matches input symbols to output symbols. The resulting functional descriptions can then be given to the FUF/SURGE surface realizer, and the sentence string it produces can be lexically compared to the original sentence in various ways to determine how well the surface realizer performs at sentence generation.

3 Implementation

The implementation necessary for evaluating the coverage of FUF/SURGE comprised three processes: (1) normalizing the syntactic/semantic representations, (2) transforming the normalizations into functional descriptions, and (3) generating the sentence itself with a surface realizer. The normalizing phase is necessary to convert the original Penn TreeBank structures into a LISP-readable format (Figure 2), which was accomplished with a series of regular expression transformations on the original text file.

The most time-consuming aspect of the procedure was creating the transformation component, which was highly analogous to writing parsing rules by hand. The resulting component contained 4000 lines of code and approximately 900 rules, although most of the actual computational effort was spent instead in surface realization. Most of the problems encountered were the result of differences in the underlying grammars themselves. For example, the Penn TreeBank has a more hierarchical noun phrase structure than the flatter representation of SURGE's systemic functional grammar.

The final task involved changing the surface realization component (1) to add additional branches and surface forms to the grammar that were not originally present in order to produce surface forms not previously possible, (2) to add new punctuation and capitalization rules[1], and (3) to update irreg-

[1]The Penn TreeBank, because it is a newspaper corpus, contains many newspaper headlines and stock quotes with domain-specific

Test Set:	Sentences	Average Time	Coverage	Close Match	Exact Match	Combined	NIST SSA
Callaway WSJ24	1346	4.15 sec.	96.1%	29.8%	54.0%	83.8%	0.9231
Callaway WSJ23	2416	3.48 sec.	98.7%	30.9%	49.0%	80.1%	0.8884
Langkilde WSJ23	2416	27.1–55.5 sec.	76.2–83.3%	N/A	5.2–57.5%	N/A	0.9450

Table 1: Comparison with HALOGEN [Langkilde, 2002]

ular morphology due to the vast number of words the system had not previously seen. The principal linguistic problems uncovered by this phase include:

- *Quotations*: Newspaper text generally contains large amounts of complex quotations, such as splitting a quoted phrase to insert the speaker in the middle, or merging a quote into an unquoted part of the sentence: "I have this feeling that it's built on sand," she says, that the market rises "but there's no foundation to it."
- *Punctuation scoping*: Problems related to the use of punctuation with tree structures[Doran, 1998]. For example, SURGE has a flat representation for noun phrases, causing difficulties with phrases such as *The major "circuit breakers"* where SURGE cannot insert punctuation between the adjective and nominal classifier.
- *Adverb and clause ordering*: Because satellite clauses in SURGE are placed using semantic information, they sometime appear in different (though still grammatically acceptable) positions than were specified in the original sentence [Elhadad *et al.*, 2001]. This can oftentimes cause a perfectly acceptable sentence to be produced, but highly skew automatic measurements of correctness such as tree edit distance (simple string accuracy). For example, contrast: "Exports fell 29% in the first few months" vs. "In the first few months, exports fell 29%."
- *Semantic roles*: SURGE has a hybrid syntactic/semantic representation, whereas the Penn TreeBank is purely syntactic. Thus some guessing must be done to fill in semantic roles in the corresponding functional description. Wrong guesses can lead to incorrect surface forms even when the remainder of the transformation was accomplished successfully.

While the normalization process is slow (10-12 hours each for WSJ23 and WSJ24), it occurs offline and only once. The transformation process is quite fast, requiring on average 0.12 seconds per sentence. Meanwhile, the FUF/SURGE system is relatively slow, as it requires the use of functional unification, a task of inherent complexity due to backtracking. Section 23 needed 8,397.2 seconds to generate 2372 sentences (with a longest exact match of 48 words), while section 24 needed 5,590.4 seconds to generate 1,326 sentences (with a longest exact match of 44), for a combined average of 3.72 seconds per sentence.[2] This contrasts with statistical approaches like Langkilde's, which require 27.1–55.5 seconds depending on algorithm parameters, and gets exponentially worse if it uses trigrams or larger models in an attempt to improve quality.

formatting not typically used in NLG systems, such as: "8 13/16% high, 8 1/2% low, 8 5/8% near closing bid, 8 3/4% offered."

[2]Additionally, a compiled C version of FUF can produce sentences using the same grammar on the order of 0.1 seconds.

4 Experiments and Results

In order to evaluate the coverage of the SURGE grammar, we used the standard train and test methodology. Unlike typical machine learning experiments, adjustments to the transformation rule set were done by hand, although the evaluation of the resulting sentences was performed automatically. Training took place over a period of several months, consisting of multiple iterations over Penn TreeBank Sections 0–22 and 24 to both improve the number of sentences which could be generated and to match as closely as possible the original sentences. These two goals were accomplished solely by adding rules to the transformation set and by updating SURGE grammar rules, notably aspects pertaining to the stock market domain, in addition to support for extended quotations which was added in previous work [Callaway and Lester, 2002].

We considered two types of string matches: exact matches that were identical character-by-character, and "close" matches which were two words or less longer or shorter than the original sentence. There were several motivations for this second choice: (1) many sentences were equivalent except for a minor missing/extra punctuation mark or wrongly capitalized word (especially with newspaper headlines); (2) as mentioned previously, movable clauses (so-called circumstantials in SURGE nomenclature) could be put in multiple acceptable locations; and (3) sentences with almost the exact number of words, especially sentences with more than 15 words, were much more likely to at least have all of the various phrases present when they were within two words or the original sentence's length. We utilized the NIST Simple String Accuracy (SSA) as an automatic evaluation score (the same as used in Langkilde's work), where the smallest number of Adds, Deletions, and Insertions were used to calculate accuracy: 1 - (A + D + T) / #Characters.

The only previous measure of generation coverage for Section 23 of the Penn TreeBank is that of [Langkilde-Geary, 2002], who defined coverage as the number of sentences for which the surface realizer produced strings. As seen in Table 1, our system achieved 96.1% on one of the training sets and 98.7% on the test set compared to between 76.2% and 83.3% for HALOGEN depending on its algorithm parameters.

A more detailed examination of the coverage and accuracy of the system is found in Table 2 for WSJ Section 24 and Table 3 for Section 23. Both tables are broken down by sentence length, which shows that the results are highly skewed towards sentence of smaller length, as would be expected in a test for exact matches. It should be noted, however, that surface realizers are rarely called upon to generate sentences

Length	# Sentences	Valid FD	Valid String	Close String	Exact Match	Combined	NIST SSA
36+	117	116 (99.2%)	103 (88.0%)	45 (38.5%)	15 (12.8%)	51.3%	0.7980
35–31	113	113 (100%)	106 (93.8%)	40 (35.4%)	26 (23.0%)	58.4%	0.8304
30–26	319	187 (99.5%)	178 (94.7%)	83 (44.2%)	61 (32.5%)	76.6%	0.8988
25–21	238	237 (99.6%)	229 (96.2%)	90 (37.8%)	114 (47.9%)	85.7%	0.9202
20–16	256	255 (100%)	246 (96.5%)	80 (31.4%)	148 (58.4%)	89.4%	0.9448
15–11	223	224 (100%)	222 (99.1%)	43 (19.2%)	174 (77.7%)	96.9%	0.9666
10– 6	152	152 (100%)	151 (99.3%)	17 (11.2%)	133 (87.5%)	98.7%	0.9831
5– 1	59	59 (100%)	59 (100%)	3 (5.1%)	56 (94.9%)	100%	0.9852
All	1346	1343 (99.8%)	1294 (96.1%)	401 (29.8%)	727 (54.0%)	83.8%	0.9231

Table 2: Sentence coverage/accuracy for the training WSJ24 sentences grouped by word length

Length	# Sentences	Valid FD	Valid String	Close String	Exact Match	Combined	NIST SSA
36+	159	158 (99.4%)	145 (91.2%)	47 (29.6%)	22 (13.8%)	43.4%	0.6607
35–31	202	199 (98.5%)	199 (98.5%)	82 (40.6%)	39 (19.3%)	59.9%	0.8238
30–26	319	317 (99.4%)	317 (99.4%)	131 (41.1%)	86 (27.0%)	68.0%	0.8518
25–21	409	405 (99.0%)	405 (99.0%)	162 (39.6%)	151 (36.9%)	76.5%	0.8725
20–16	468	465 (98.9%)	465 (98.9%)	173 (36.8%)	230 (48.9%)	85.7%	0.9097
15–11	440	437 (99.5%)	437 (99.5%)	106 (24.2%)	289 (65.8%)	90.0%	0.9313
10– 6	279	279 (100%)	279 (100%)	36 (12.9%)	238 (85.3%)	98.2%	0.9673
5– 1	141	140 (99.3%)	140 (99.3%)	11 (7.8%)	129 (91.5%)	99.3%	0.9826
All	2416	2400 (99.3%)	2387 (98.7%)	748 (30.9%)	1184 (49.0%)	79.9%	0.8884

Table 3: Sentence coverage/accuracy for the unseen WSJ23 sentences grouped by word length

with the extended lengths and complexities found in highly educated newspaper text. Finally, the "valid FD" column indicates that the transformation program is very good at producing valid functional descriptions, even if they eventually are discarded by the grammar as being erroneous.

The test set had a slightly higher coverage than the training set shown above, although a lower number of perfect matches and a lower score using the NIST SSA measure. Although we trained on other sections of the Penn TreeBank, time constraints due to the large amount of time required to generate all test sentences prevented us from having a fuller comparison set. Additionally, Section 24 was the first section we trained on, and it is likely that later sections were more similar to Section 23, or that the amount of domain-specific stock market constructions were imbalanced. Finally, the "close matches" column shows how many candidate sentences might be nearly exactly matching, and the sum of these two is reflected in the final category "combined."

One interesting observation is that this evaluation (and correspondingly, the evaluation of HALOGEN) is not only an evaluation of the underlying surface realizer, but also of the accompanying transformation program that converts the Penn TreeBank notation into the specifications it expects. We thus set out to perform a minor, secondary evaluation to determine if it were possible to find a baseline metric for how many sentences could still be generated by the surface realizer even if the corresponding FDs could not be produced for them by the transformation program.

We thus randomly selected 25 sentences from each of the two sections which were not either perfect matches or "close" matches, *i.e.*, they were not in the "combined" match category. While these sets included some of the problems listed in Section 3, 41 of the 50 were capable of being rendered by hand as FDs which produced exact matches without changes to the grammar, 6 required minor changes to the grammar which were quickly performed, and the remaining 3 sentences required major grammar changes which have still not yet been made. The latter were sentences that still do not have satisfactory linguistic analyses in the linguistic literature. We thus conclude that with better transformation rules, we could then obtain close to 95% coverage.

5 Discussion

Surface realization is probably the most understood and competent task in NLG today. There is a high possibility that surface realization can already be considered a solved problem, except with regard to problems introduced by new languages or highly specialized domains. However, there are two related unsolved problems inherent in the process described in this paper.

5.1 Automatic Evaluation of Output

Evaluation of NLG systems face the same problems as those that confront Machine Translation systems: Given a set of generated sentences, how do you tell how "good" they are in general, and how often you can produce good sentences in a given context or application. Work in machine translation has shifted to large-scale evaluations which require automatic evaluation techniques [Papineni *et al.*, 2001;

Doddington, 2002] because human graders cannot hope to examine all of the responses in a short enough period of time.

Yet current evaluation techniques are completely numeric/statistical in nature and do not attempt to measure semantic content (such as the well-known example of a missing "not" in a system's output). Furthermore, these techniques are ill-equipped to evaluate the types of sentences produced by symbolic NLG systems. For example, by changing a feature specifying that, say, a particularly lengthy purpose clause should go at the beginning rather than the end of a sentence, a string edit distance metric will report a very large error when in terms of the system's input, only one "error" has occurred.

As an example of this type of problem, consider that the string edit distance between the following original sentence and that produced by the transformation program and FUF/SURGE with input from the Penn TreeBank would be 83, resulting in a NIST SSA 50.4% match:

> Freddie Mac said the principal-only securities were priced at 58 1/4 to yield 8.45%, **assuming an average life of eight years and a prepayment of 160% of the PSA model.**
>
> Freddie Mac said **assuming an average life of eight years and a prepayment of 160% of the PSA model**, the principal-only securities were priced at 58 1/4 to yield 8.45%.

Obviously 50.4% is a poor score for such sentences, but many such examples were found in our corpus and their low scores were factored into the NIST Simple String Accuracy ratios in Tables 2 and 3.

5.2 Symbolic vs. Statistical Approaches

Almost all fully-developed NLG systems to-date operate on data specified in a knowledge base from some other system. The fact that this data has typically been represented as highly-structured data has been an impediment to traditional machine learning techniques which have previously operated mostly on flat, unstructured text data. It is also generally stated that statistical methods are more robust than their symbolic counterparts and more easily adapted to new data sets. The data in the previous section seems to indicate that HALOGEN, a statistical system, performs substantially better on longer sentences, even if it has lower overall coverage. But there are several advantages in favor of symbolic techniques.

First, the transformation program presented above can be tweaked to an arbitrary level of perfection by progressively adding more rules. Most statistical and machine learning systems however have eventually reached a boundary where progress becomes seemingly exponentially more difficult. Second, errors which are encountered during processing can be examined and fixed because the grammars and other resources are logically and semantically connected to the language being generated, rather than being a set of numbers. If however a statistical generator must create new output forms not contained in the initial corpus or model, it must be retrained from scratch.

Finally, symbolic surface realizers allow a wide range of optional operations which statistical programs currently can't offer, for example, the capability of adding formatting statements in HTML, modifying punctuation, generating dialect differences, adding prosody for TTS, *etc*. While this may be due to the multiple decades of history over which symbolic systems have been developed, it may also be due to the lack of annotated corpora that support statistical algorithms, or even potentially the impossibility of having a corpus at all, such as in large narratives [Callaway and Lester, 2002].

Additionally, there are some disadvantages to the current approaches undertaken in statistical NLG research. For instance, symbolic NL generation systems are already considered slow, and FUF/SURGE is generally considered to be the slowest in the NLG community. And yet the data from Table 3 shows that HALOGEN is anywhere from 6.5 to 16 times slower, and thus a 10-sentence paragraph might need 4 minutes or longer to be generated. Moreover, these approaches use techniques such as n-gram models, where n must be increased to improve quality, but results in even slower generation times and exponentially larger storage space.

5.3 Potential Applications

The transformation program presented here has additional side benefits besides helping calculate the coverage of a grammar. For example, in generation systems where non-linguists must maintain old data and add new data, such a program allows them to write sample sentences in the syntax-only TreeBank notation, which is much easier for non-linguists, and then convert those sentences directly into a more linguistically-manageable form for generation (*e.g.*, functional descriptions). Graphical editing tools for linguistic data such as GATE [Bontcheva *et al.*, 2002] or similar authoring tools could quicken the process even more.

Additionally, the transformation program can be used as an error-checker for the well-formedness of sentences contained in the TreeBank. Rules could be (and have been) added alongside the normal transformation rules that detect when errors are encountered, categorize them, and make them available to the corpus creator for correction. This extends not only to annotation errors by the corpus creator detectable at the syntax tree level, but even morphology errors such as incorrect verbs, typos, or British/American English differences by the original author of the text. Both of these tasks are much more difficult for a statistical system to accomplish, requiring separate retraining in the first case and locating or creating a corpus of possible mistakes in the second, much like what is done with tutoring systems where databases of potential student errors are painstakingly constructed.

6 Conclusions and Future Work

Recent years have seen the arrival of statistical approaches to the field of Natural Language Generation, much as was seen in parsing a decade ago. Of the many possible components in the standard NLG pipelined architecture, almost all of these statistical systems have focused on the surface realization component, offering the same robustness, wide coverage, and domain- and language-independence as for parsing.

Recent experiments with one statistically-based system, HALOGEN, showed that it could achieve respectable coverage without the typical development costs inherent in producing grammars for symbolic NLG. By taking as input sentences from the Penn TreeBank, HALOGEN was able to generate a substantial enough quantity of sentences to allow for an empirical analysis of grammatical coverage of English. This paper represents the analogous effort for a symbolic generation system using the FUF/SURGE systemic realization system, which includes the largest generation grammar.

We presented the results of a grammatical coverage evaluation experiment that showed the symbolic system had a higher level of coverage of English as represented by the Penn TreeBank. We also contrasted the statistical and symbolic methodologies and concluded with ideas for future applications for easier creation of NLG systems and automatic error-checking for large-scale corpora. We also plan to further develop our Italian generation grammar developed at ITC-irst [Novello and Callaway, 2003] with a similarly annotated corpus of Italian newspaper text.

7 Acknowledgements

This work was funded by the PEACH project, granted by the Autonomous Province of Trento in Italy.

References

[Bangalore and Rambow, 2000] Srinivas Bangalore and Owen 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.

[Bontcheva *et al.*, 2002] K. Bontcheva, H. Cunningham, V. Tablan, D. Maynard, and H. Saggion. Development of reusable and robust language processing components for information systems using gate. In *Proceedings of the 3rd International Workshop on Natural Language and Information Systems*, 2002.

[Callaway and Lester, 2002] Charles B. Callaway and James C. Lester. Narrative prose generation. *Artificial Intelligence*, 139(2):213–252, 2002.

[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.

[Elhadad *et al.*, 2001] Michael Elhadad, Yael Netzer, Regina Barzilay, and Kathleen McKeown. Ordering circumstantials for multi-document summarization. In *Proceedings of the Bar-Ilan Symposium on the Foundations of Artificial Intelligence*, Jerusalem, Israel, June 2001.

[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 Association for Computational Linguistics and the 17th International Conference on Computational Linguistics*, 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*, pages 17–24, 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.

[Novello and Callaway, 2003] Alessandra Novello and Charles Callaway. Porting to an Italian surface realizer: A case study. In *Proceedings of the 9th European Workshop on NLG*, Budapest, Hungary, April 2003.

[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 Division, T. J. Watson Research Center, New York, September 2001.

[Pollard and Sag, 1994] C. Pollard and I. Sag. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press, Chicago, 1994.

[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.

[Reiter, 1994] Ehud Reiter. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the Seventh International Workshop on Natural Language Generation*, pages 163–170, Kennebunkport, ME, 1994.

[Robin and McKeown, 1995] Jacques Robin and Kathy McKeown. Empirically designing and evaluating a new revision-based model for summary generation. *Artificial Intelligence*, 85(1–2), 1995.

[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.