Evaluation Metrics for Generation

Srinivas Bangalore and **Owen Rambow** and **Steve Whittaker**
AT&T Labs – Research
180 Park Ave, PO Box 971
Florham Park, NJ 07932-0971, USA
{srini,rambow,stevew}@research.att.com

Abstract

Certain generation applications may profit from the use of stochastic methods. In developing stochastic methods, it is crucial to be able to quickly assess the relative merits of different approaches or models. In this paper, we present several types of intrinsic (system internal) metrics which we have used for baseline quantitative assessment. This quantitative assessment should then be augmented to a fuller evaluation that examines qualitative aspects. To this end, we describe an experiment that tests correlation between the quantitative metrics and human qualitative judgment. The experiment confirms that intrinsic metrics cannot replace human evaluation, but some correlate significantly with human judgments of quality and understandability and can be used for evaluation during development.

1 Introduction

For many applications in natural language generation (NLG), the range of linguistic expressions that must be generated is quite restricted, and a grammar for a surface realization component can be fully specified by hand. Moreover, in many cases it is very important not to deviate from very specific output in generation (e.g., maritime weather reports), in which case hand-crafted grammars give excellent control. In these cases, evaluations of the generator that rely on human judgments (Lester and Porter, 1997) or on human annotation of the test corpora (Kukich, 1983) are quite sufficient.

However, in other NLG applications the variety of the output is much larger, and the demands on the quality of the output are somewhat less stringent. A typical example is NLG in the context of (interlingua- or transfer-based) machine translation. Another reason for relaxing the quality of the output may be that not enough time is available to develop a full grammar for a new target language in NLG. In all these cases, stochastic methods provide an alternative to hand-crafted approaches to NLG. To our knowledge, the first to use stochastic techniques in an NLG realization module were Langkilde and Knight (1998a) and (1998b) (see also (Langkilde, 2000)). As is the case for stochastic approaches in natural language understanding, the research and development itself requires an effective intrinsic metric in order to be able to evaluate progress.

In this paper, we discuss several evaluation metrics that we are using during the development of FERGUS (Flexible Empiricist/Rationalist Generation Using Syntax). FERGUS, a realization module, follows Knight and Langkilde's seminal work in using an n-gram language model, but we augment it with a tree-based stochastic model and a lexicalized syntactic grammar. The metrics are useful to us as relative quantitative assessments of different models we experiment with; however, we do not pretend that these metrics in themselves have any validity. Instead, we follow work done in dialog systems (Walker et al., 1997) and attempt to find metrics which on the one hand can be computed easily but on the other hand correlate with empirically verified human judgments in qualitative categories such as readability.

The structure of the paper is as follows. In Section 2, we briefly describe the architecture of FERGUS, and some of the modules. In Section 3 we present four metrics and some results obtained with these metrics. In Section 4 we discuss the for experimental validation of the metrics using human judgments, and present a new metric based on the results of these experiments. In Section 5 we discuss some of the many problematic issues related to the use of metrics and our metrics in particular, and discuss on-going work.

2 System Overview

FERGUS is composed of three modules: the Tree Chooser, the Unraveler, and the Linear Precedence (LP) Chooser (Figure 1). The input to the system is

a dependency tree as shown in Figure 2.[1] Note that the nodes are unordered and are labeled only with lexemes, not with any sort of syntactic annotations.[2] The Tree Chooser uses a stochastic tree model to choose syntactic properties (expressed as trees in a Tree Adjoining Grammar) for the nodes in the input structure. This step can be seen as analogous to "supertagging" (Bangalore and Joshi, 1999), except that now supertags (i.e., names of trees which encode the syntactic properties of a lexical head) must be found for words in a tree rather than for words in a linear sequence. The Tree Chooser makes the simplifying assumptions that the choice of a tree for a node depends only on its daughter nodes, thus allowing for a top-down algorithm. The Tree Chooser draws on a tree model, which is a analysis in terms of syntactic dependency for 1,000,000 words of the Wall Street Journal (WSJ).[3]

The supertagged tree which is output from the Tree Chooser still does not fully determine the surface string, because there typically are different ways to attach a daughter node to her mother (for example, an adverb can be placed in different positions with respect to its verbal head). The Unraveler therefore uses the XTAG grammar of English (XTAG-Group, 1999) to produce a lattice of all possible linearizations that are compatible with the supertagged tree. Specifically, the daughter nodes are ordered with respect to the head at each level of the derivation tree. In cases where the XTAG grammar allows a daughter node to be attached at more than one place in the mother supertag (as is the case in our example for *was* and *for; generally, such underspecification occurs with adjuncts and with arguments if their syntactic role is not specified*), a disjunction of all these positions is assigned to the daughter node. A bottom-up algorithm then constructs a lattice that encodes the strings represented by each level of the derivation tree. The lattice at the root of the derivation tree is the result of the Unraveler.

Finally, the LP Chooser chooses the most likely traversal of this lattice, given a linear language model (n-gram). The lattice output from the Unraveler encodes all possible word sequences permitted by the supertagged dependency structure. We rank these word sequences in the order of their likelihood by composing the lattice with a finite-state machine representing a trigram language model. This model has been constructed from the 1,000,0000

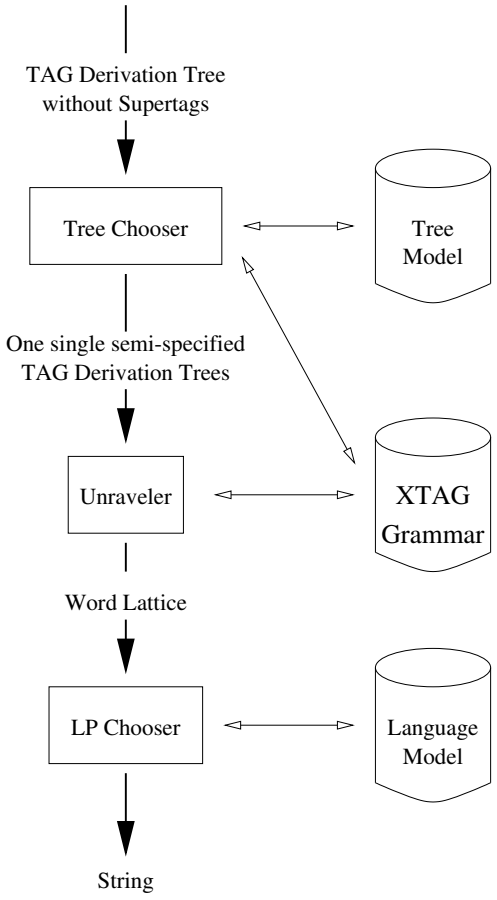

Figure 1: Architecture of FERGUS

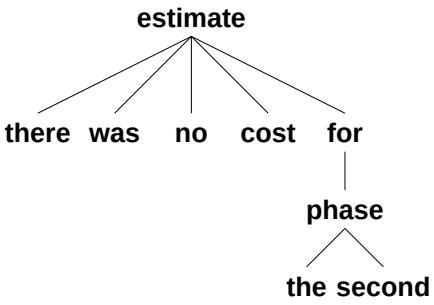

Figure 2: Input to FERGUS

[1] The sentence generated by this tree is a predicative noun construction. The XTAG grammar analyzes these as being headed by the noun rather than by the copula, and we follow the XTAG analysis. However, it would of course also be possible to use a grammar that allows for the copula-headed analysis.

[2] In the system that we used in the experiments described in Section 3, all words (including function words) need to be present in the input representation, fully inflected. Furthermore, there is no indication of syntactic role at all. This is of course unrealistic for applications – see Section 5 for further remarks.

[3] This was constructed from the Penn Tree Bank using some heuristics, since the Penn Tree Bank does not contain full head-dependent information; as a result of the use of heuristics, the Tree Model is not fully correct.

words WSJ training corpus. We pick the best path through the lattice resulting from the composition using the Viterbi algorithm, and this top ranking word sequence is the output of the LP Chooser and the generator.

3 Baseline Quantitative Metrics

We have used four different baseline quantitative metrics for evaluating our generator. The first two metrics are based entirely on the surface string. The next two metrics are based on a syntactic representation of the sentence.

3.1 String-Based Metrics

We employ two metrics that measure the accuracy of a generated string. The first metric, **simple accuracy**, is the same string distance metric used for measuring speech recognition accuracy. This metric has also been used to measure accuracy of MT systems (Alshawi et al., 1998). It is based on string edit distance between the output of the generation system and the reference corpus string. Simple accuracy is the number of insertion (I), deletion (D) and substitutions (S) errors between the reference strings in the test corpus and the strings produced by the generation model. An alignment algorithm using substitution, insertion and deletion of tokens as operations attempts to match the generated string with the reference string. Each of these operations is assigned a cost value such that a substitution operation is cheaper than the combined cost of a deletion and an insertion operation. The alignment algorithm attempts to find the set of operations that minimizes the cost of aligning the generated string to the reference string. The metric is summarized in Equation (1). R is the number of tokens in the target string.

(1) **Simple String Accuracy** $= (1 - \frac{I+D+S}{R})$

Consider the following example. The target sentence is on top, the generated sentence below. The third line represents the operation needed to transform one sentence into another; a period is used to indicate that no operation is needed.[4]

(2) There was no cost estimate for the
There was estimate for phase the
. . d d . . i .
second phase
second no cost
. i s

[4]Note that the metric is symmetric.

When we tally the results we obtain the score shown in the first column of Table 1.

Note that if there are insertions and deletions, the number of operations may be larger than the number of tokens involved for either one of the two strings. As a result, the simple string accuracy metric may be negative (though it is never greater than 1, of course).

The simple string accuracy metric penalizes a misplaced token twice, as a deletion from its expected position and insertion at a different position. This is particularly worrisome in our case, since in our evaluation scenario the generated sentence is a permutation of the tokens in the reference string. We therefore use a second metric, **Generation String Accuracy**, shown in Equation (3), which treats deletion of a token at one location in the string and the insertion of the same token at another location in the string as one single movement error (M). This is in addition to the remaining insertions (I') and deletions (D').

(3) **Generation String Accuracy** $= (1 - \frac{M+I'+D'+S}{R})$

In our example sentence (2), we see that the insertion and deletion of *no* can be collapsed into one move. However, the wrong positions of *cost* and of *phase* are not analyzed as two moves, since one takes the place of the other, and these two tokens still result in one deletion, one substitution, and one insertion.[5] Thus, the generation string accuracy depenalizes simple moves, but still treats complex moves (involving more than one token) harshly. Overall, the scores for the two metrics introduced so far are shown in the first two columns of Table 1.

3.2 Tree-Based Metrics

While the string-based metrics are very easy to apply, they have the disadvantage that they do not reflect the intuition that all token moves are not equally "bad". Consider the subphrase *estimate for phase the second* of the sentence in (2). While this is bad, it seems better than an alternative such as *estimate phase for the second.* The difference between the two strings is that the first scrambled string, but not the second, can be read off from the dependency tree for the sentence (as shown in Figure 2) without violation of projectivity, i.e., without (roughly

[5]This shows the importance of the alignment algorithm in the definition of these two metrics: had it not aligned *phase* and *cost* as a substitution (but each with an empty position in the other string instead), then the simple string accuracy would have 6 errors instead of 5, but the generation string accuracy would have 3 errors instead of 4.

speaking) creating discontinuous constituents. It has long been observed (though informally) that the dependency trees of a vast majority of sentences in the languages of the world are projective (see e.g. (Mel'čuk, 1988)), so that a violation of projectivity is presumably a more severe error than a word order variation that does not violate projectivity.

We designed the **tree-based accuracy** metrics in order to account for this effect. Instead of comparing two strings directly, we relate the two strings to a dependency tree of the reference string. For each treelet (i.e., non-leaf node with all of its daughters) of the reference dependency tree, we construct strings of the head and its dependents in the order they appear in the reference string, and in the order they appear in the result string. We then calculate the number of substitutions, deletions, and insertions as for the simple string accuracy, and the number of substitutions, moves, and remaining deletions and insertions as for the generation string metrics, for all treelets that form the dependency tree. We sum these scores, and then use the values obtained in the formulas given above for the two string-based metrics, yielding the **Simple Tree Accuracy** and **Generation Tree Accuracy**. The scores for our example sentence are shown in the last two columns of Table 1.

3.3 Evaluation Results

Here we summarize two experiments that we have performed that use different tree models. (For a more detailed comparisons of different tree models, see (Bangalore and Rambow, 2000).)

- For the baseline experiment, we impose a random tree structure for each sentence of the corpus and build a Tree Model whose parameters consist of whether a lexeme l_d precedes or follows her mother lexeme l_m. We call this the Baseline Left-Right (LR) Model. This model generates *There was estimate for phase the second no cost .* for our example input.

- In the second experiment we use the system as described in Section 2. We employ the supertag-based tree model whose parameters consist of whether a lexeme l_d with supertag s_d is a dependent of lexeme l_m with supertag s_m. Furthermore we use the information provided by the XTAG grammar to order the dependents. This model generates *There was no cost estimate for the second phase .* for our example input, which is indeed the sentence found in the WSJ.

The simple accuracy, generation accuracy, simple tree accuracy and generation tree accuracy for the two experiments are tabulated in Table 2. The test corpus is a randomly chosen subset of 100 sentences from the Section 20 of WSJ. The dependency structures for the test sentences were obtained automatically from converting the Penn TreeBank phrase structure trees, in the same way as was done to create the training corpus. The average length of the test sentences is 16.7 words with a longest sentence being 24 words in length. As can be seen, the supertag-based model improves over the baseline LR model on all four baseline quantitative metrics.

4 Qualitative Evaluation of the Quantitative Metrics

4.1 The Experiments

We have presented four metrics which we can compute automatically. In order to determine whether the metrics correlate with independent notions understandability or quality, we have performed evaluation experiments with human subjects.

In the web-based experiment, we ask human subjects to read a short paragraph from the WSJ. We present three or five variants of the last sentence of this paragraph on the same page, and ask the subject to judge them along two dimensions:

- **Understandability**: How easy is this sentence to understand? Options range from "Extremely easy" (= 7) to "Just barely possible" (=4) to "Impossible" (=1). (Intermediate numeric values can also be chosen but have no description associated with them.)
- **Quality**: How well-written is this sentence? Options range from "Extremely well-written" (= 7) to "Pretty bad" (=4) to "Horrible (=1). (Again, intermediate numeric values can also be chosen, but have no description associated with them.)

The 3-5 variants of each of 6 base sentences are constructed by us (most of the variants have not actually been generated by FERGUS) to sample multiple values of each intrinsic metric as well as to contrast differences between the intrinsic measures. Thus for one sentence "tumble", two of the five variants have approximately identical values for each of the metrics but with the absolute values being high (0.9) and medium (0.7) respectively. For two other sentences we have contrasting intrinsic values for tree and string based measures. For the final sentence we have contrasts between the string measures with

Metric	Simple String Accuracy	Generation String Accuracy	Simple Tree Accuracy	Generation Tree Accuracy
Total number of tokens	9	9	9	9
Unchanged	6	6	6	6
Substitutions	1	1	0	0
Insertions	2	1	3	0
Deletions	2	1	3	0
Moves	0	1	0	3
Total number of problems	5	4	6	3
Score	0.44	0.56	0.33	0.67

Table 1: Scores for the sample sentence according to the four metrics

Tree Model	Simple String Accuracy	Generation String Accuracy	Simple Tree Accuracy	Generation Tree Accuracy
Baseline LR Model	0.41	0.56	0.41	0.63
Supertag-based Model	0.58	0.72	0.65	0.76

Table 2: Performance results from the two tree models

tree measures being approximately equal. Ten subjects who were researchers from AT&T carried out the experiment. Each subject made a total of 24 judgments.

Given the variance between subjects we first normalized the data. We subtracted the mean score for each subject from each observed score and then divided this by standard deviation of the scores for that subject. As expected our data showed strong correlations between normalized understanding and quality judgments for each sentence variant ($r_{(22)} = 0.94$, $p < 0.0001$).

Our main hypothesis is that the two tree-based metrics correlate better with both understandability and quality than the string-based metrics. This was confirmed. Correlations of the two string metrics with normalized understanding for each sentence variant were not significant ($r_{(22)} = 0.08$ and $r_{(22)} = 0.23$, for simple accuracy and generation accuracy: for both $p > 0.05$). In contrast both of the tree metrics were significant ($r_{(22)} = 0.51$ and $r_{(22)} = 0.48$: for tree accuracy and generation tree accuracy, for both $p < 0.05$). Similar results were achieved for the normalized quality metric: ($r_{(22)} = 0.16$ and $r_{(22)} = 0.33$: for simple accuracy and generation accuracy, for both $p > 0.05$), ($r_{(22)} = 0.45$ and $r_{(22)} = 0.42$, for tree accuracy and generation tree accuracy, for both $p < 0.05$).

A second aim of our qualitative evaluation was to test various models of the relationship between intrinsic variables and qualitative user judgments. We proposed a number of models in which various combinations of intrinsic metrics were used to predict user judgments of understanding and quality. We conducted a series of linear regressions with normalized judgments of understanding and quality as the dependent measures and as independent measures different combinations of one of our four metrics with sentence length, and with the "problem" variables that we used to define the string metrics (S, I, D, M, I', D' – see Section 3 for definitions). One sentence variant was excluded from the data set on the grounds that the severely "mangled" sentence happened to turn out well-formed and with nearly the same meaning as the target sentence. The results are shown in Table 3.

We first tested models using one of our metrics as a single intrinsic factor to explain the dependent variable. We then added the "problem" variables,[6] and could boost the explanatory power while maintaining significance. In Table 3, we show only some combinations, which show that the best results were obtained by combining the simple tree accuracy with the number of Substitutions (S) and the sentence length. As we can see, the number of substitutions has an important effect on explanatory power, while that of sentence length is much more modest (but more important for quality than for understanding). Furthermore, the number of substitutions has more explanatory power than the number of moves (and in fact than any of the other "problem" variables).

The two regressions for understanding and writing show very similar results. Normalized understand-

[6]None of the "problem" variables have much explanatory power on their own (nor did they achieve significance).

Model	User Metric	Explanatory Power (R^2)	Statistical Significance (p value)
Simple String Accuracy	Understanding	0.02	0.571
Simple String Accuracy	Quality	0.00	0.953
Generation String Accuracy	Understanding	0.02	0.584
Generation String Accuracy	Quality	0.05	0.327
Simple Tree Accuracy	Understanding	0.36	0.003
Simple Tree Accuracy	Quality	0.34	0.003
Generation Tree Accuracy	Understanding	0.35	0.003
Generation Tree Accuracy	Quality	0.35	0.003
Simple Tree Accuracy + S	Understanding	0.48	0.001
Simple Tree Accuracy + S	Quality	0.47	0.002
Simple Tree Accuracy + M	Understanding	0.38	0.008
Simple Tree Accuracy + M	Quality	0.34	0.015
Simple Tree Accuracy + Length	Understanding	0.40	0.006
Simple Tree Accuracy + Length	Quality	0.42	0.006
Simple Tree Accuracy + S + Length	Understanding	0.51	0.003
Simple Tree Accuracy + S + Length	Quality	0.53	0.002

Table 3: Testing different models of user judgments (S is number of substitutions, M number of moved elements)

ing was best modeled as:

> Normalized understanding = 1.4728*simple tree accuracy - 0.1015*substitutions - 0.0228 * length - 0.2127.

This model was significant: $F_{(3,19)} = 6.62$, $p < 0.005$. The model is plotted in Figure 3, with the data point representing the removed outlier at the top of the diagram.

This model is also intuitively plausible. The simple tree metric was designed to measure the quality of a sentence and it has a positive coefficient. A substitution represents a case in the string metrics in which not only a word is in the wrong place, but the word that should have been in that place is somewhere else. Therefore, substitutions, more than moves or insertions or deletions, represent grave cases of word order anomalies. Thus, it is plausible to penalize them separately. (Note that the simple tree accuracy is bounded by 1, while the number of substitutions is bounded by the length of the sentence. In practice, in our sentences S ranges between 0 and 10 with a mean of 1.583.) Finally, it is also plausible that longer sentences are more difficult to understand, so that length has a (small) negative coefficient.

We now turn to model for quality.

> Normalized quality = 1.2134*simple tree accuracy - 0.0839*substitutions - 0.0280 * length - 0.0689.

This model was also significant: $F_{(3,19)} = 7.23$, $p < 0.005$. The model is plotted in Figure 4, with the data point representing the removed outlier at the top of the diagram. The quality model is plausible for the same reasons that the understanding model is.

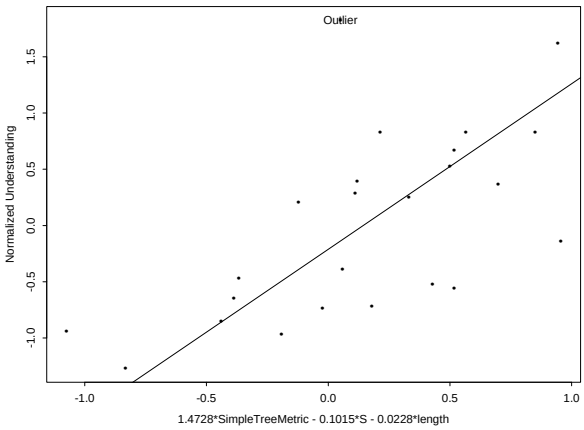

Figure 3: Regression for Understanding

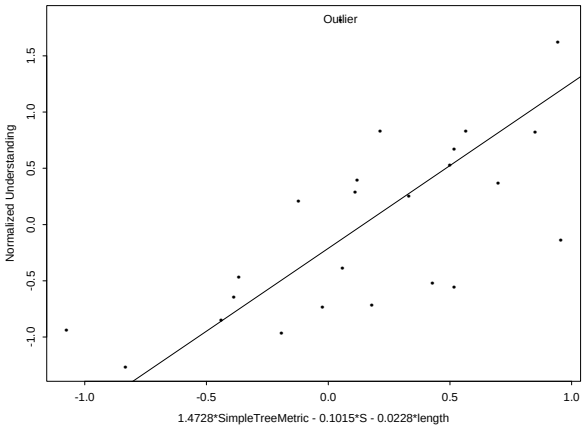

Figure 4: Regression for Quality (Well-Formedness)

4.2 Two New Metrics

A further goal of these experiments was to obtain one or two metrics which can be automatically computed, and which have been shown to significantly correlate with relevant human judgments. We use as a starting point the two linear models for normalized understanding and quality given above, but we make two changes. First, we observe that while it is plausible to model human judgments by penalizing long sentences, this seems unmotivated in an accuracy metric: we do not want to give a perfectly generated longer sentence a lower score than a perfectly generated shorter sentence. We therefore use models that just use the simple tree accuracy and the number of substitutions as independent variables. Second, we note that once we have done so, a perfect sentence gets a score of 0.8689 (for understandability) or 0.6639 (for quality). We therefore divide by this score to assure that a perfect sentence gets a score of 1. (As for the previously introduced metrics, the scores may be less than 0.)

We obtain the following new metrics:

(4) **Understandability Accuracy** = (1.3147*simple tree accuracy - 0.1039*substitutions - 0.4458) / 0.8689

(5) **Quality Accuracy** = (1.0192*simple tree accuracy - 0.0869*substitutions - 0.3553) / 0.6639

We reevaluated our system and the baseline model using the new metrics, in order to verify whether the more motivated metrics we have developed still show that FERGUS improves performance over the baseline. This is indeed the case; the results are summarized in Table 4.

Tree Model	Understandability Accuracy	Quality Accuracy
Baseline	-0.08	-0.12
Supertag-based	0.44	0.42

Table 4: Performance results from the two tree models using the new metrics

5 Discussion

We have devised the baseline quantitative metrics presented in this paper for internal use during research and development, in order to evaluate different versions of FERGUS. However, the question also arises whether they can be used to compare two completely different realization modules. In either case, there are two main issues facing the proposed corpus-based quantitative evaluation: does it generalize and is it fair?

The problem in generalization is this: can we use this method to evaluate anything other than versions of FERGUS which generate sentences from the WSJ? We claim that we can indeed use the quantitative evaluation procedure to evaluate most realization modules generating sentences from any corpus of unannotated English text. The fact that the tree-based metrics require dependency parses of the corpus is not a major impediment. Using existing syntactic parsers plus ad-hoc postprocessors as needed, one can create the input representations to the generator as well as the syntactic dependency trees needed for the tree-based metrics. The fact that the parsers introduce errors should not affect the way the scores are used, namely as relative scores (they have no real value absolutely). Which realization modules can be evaluated? First, it is clear that our approach can only evaluate single-sentence realization modules which may perform some sentence planning tasks, but crucially not including sentence scoping/aggregation. Second, this approach only works for generators whose input representation is fairly "syntactic". For example, it may be difficult to evaluate in this manner a generator that uses semantic roles in its input representation, since we currently cannot map large corpora of syntactic parses onto such semantic representations, and therefore cannot create the input representation for the evaluation.

The second question is that of fairness of the evaluation. FERGUS as described in this paper is of limited use, since it only chooses word order (and, to a certain extent, syntactic structure). Other realization and sentence planning tasks which are needed for most applications and which may profit from a

stochastic model include lexical choice, introduction of function words and punctuation, and generation of morphology. (See (Langkilde and Knight, 1998a) for a relevant discussion. FERGUS currently can perform punctuation and function word insertion, and morphology and lexical choice are under development.) The question arises whether our metrics will fairly measure the quality of a more complete realization module (with some sentence planning). Once the range of choices that the generation component makes expands, one quickly runs into the problem that, while the gold standard may be a good way of communicating the input structure, there are usually other good ways of doing so as well (using other words, other syntactic constructions, and so on). Our metrics will penalize such variation. However, in using stochastic methods one is of course precisely interested in learning from a corpus, so that the fact that there may be other ways of expressing an input is less relevant: the whole point of the stochastic approach is precisely to express the input in a manner that resembles as much as possible the realizations found in the corpus (given its genre, register, idiosyncratic choices, and so on). Assuming the test corpus is representative of the training corpus, we can then use our metrics to measure deviance from the corpus, whether it be merely in word order or in terms of more complex tasks such as lexical choice as well. Thus, as long as the goal of the realizer is to emulate as closely as possible a given corpus (rather than provide a maximal range of paraphrastic capability), then our approach can be used for evaluation.[7]

As in the case of machine translation, evaluation in generation is a complex issue. (For a discussion, see (Mellish and Dale, 1998).) Presumably, the quality of most generation systems can only be assessed at a system level in a task-oriented setting (rather than by taking quantitative measures or by asking humans for quality assessments). Such evaluations are costly, and they cannot be the basis of work in stochastic generation, for which evaluation is a frequent step in research and development. An advantage of our approach is that our quantitative metrics allow us to evaluate without human intervention, automatically and objectively (objectively with respect to the defined metric, that is). Independently, the use of the metrics has been validated using human subjects (as discussed in Section 4); once this has happened, the researcher can have increased confidence that choices made in research and development based on the quantitative metrics will in fact correlate with relevant subjective qualitative measures.

[7] We could also assume a set of acceptable paraphrases for each sentence in the test corpus. Our metrics are run on all paraphrases, and the best score chosen. However, for many applications it will not be easy to construct such paraphrase sets, be it by hand or automatically.

References

Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 1998. Automatic acquisition of hierarchical transduction models for machine tr anslation. In *Proceedings of the 36th Annual Meeting Association for Computational Linguistics*, Montreal, Canada.

Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2).

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbrücken, Germany.

Karen Kukich. 1983. *Knowledge-Based Report Generation: A Knowledge Engineering Approach to Natural Language Report Generation.* Ph.D. thesis, University of Pittsuburgh.

Irene Langkilde and Kevin Knight. 1998a. Generation that exploits corpus-based statistical knowledge. In *36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*, pages 704–710, Montréal, Canada.

Irene Langkilde and Kevin Knight. 1998b. The practical value of n-grams in generation. In *Proceedings of the Ninth International Natural Language Generation Workshop (INLG'98)*, Niagara-on-the-Lake, Ontario.

Irene Langkilde. 2000. Forest-based statistical sentence generation. In *6th Applied Natural Language Processing Conference (ANLP'2000)*, pages 170–177, Seattle, WA.

James C. Lester and Bruce W. Porter. 1997. Developing and empirically evaluating robust explanation generators: The KNIGHT experiments. *Computational Linguistics*, 23(1):65–102.

Igor A. Mel'čuk. 1988. *Dependency Syntax: Theory and Practice.* State University of New York Press, New York.

Chris Mellish and Robert Dale. 1998. Evaluation in the context of natural language generation. *Computer Speech and Language*, 12:349–373.

M. A. Walker, D. Litman, C. A. Kamm, and A. Abella. 1997. PARADISE: A general framework for evaluating spoken dialogue agents. In *Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics, ACL/EACL 97*, pages 271–280.

The XTAG-Group. 1999. A lexicalized Tree Adjoining Grammar for English. Technical report, Institute for Research in Cognitive Science, University of Pennsylvania.