

Evaluation Metrics for Generation

Srinivas Bangalore and Owen Rambow and Steve Whittaker
AT&T Labs – Research
rambow@research.att.com

February 3, 2000

Session: Evaluation Track

Topic Area(s): Evaluation, Tactical Generation

Keywords: Evaluation, Stochastic Generation, Sentence Realization, Syntactic Models

Word Count: 3,179

Under consideration for other conferences (specify)? None

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 two types of metrics which we have used for baseline quantitative assessment. This quantitative assessment can then be augmented to a fuller evaluation that examines qualitative aspects. To this end, we describe an experiment intended to show correlation between the quantitative metrics and human qualitative judgment. Initial data suggests that tree-based metrics correlate significantly with human judgments of quality and understandability.

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). 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 our metrics and some results obtained with these metrics. In Section 4 we discuss our plans for experimental validation of the metrics using human judgments, and report

initial results. 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. Note that the nodes are unordered and are labeled only with lexemes, not with any sort of syntactic annotations.¹ 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 dynamic programming 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).²

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 a daughter node can be attached at more than one place in the head supertag (as is the case in our example for *was* and *for*), a disjunction of all these positions are 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

¹In 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.

²This 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.

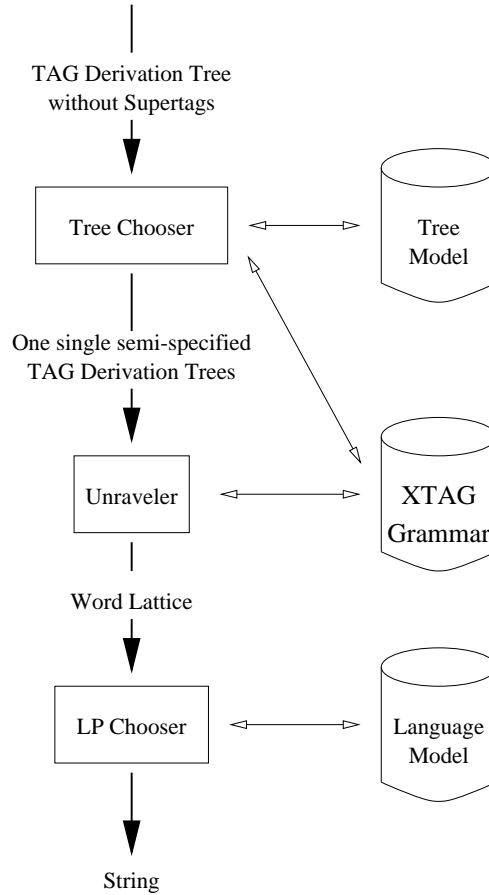


Figure 1: Architecture of FERGUS

the Unraveler. The resulting lattice for the example sentence is shown in Figure 3.

Finally, the LP Chooser chooses the most likely traversal of this lattice, given a linear language model. 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 1,000,000 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.

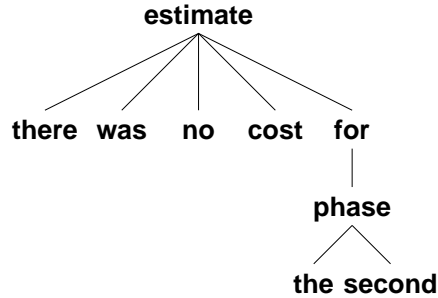


Figure 2: Input to FERGUS

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)

$$\mathbf{SimpleAccuracy} = (1 - \frac{I + D + S}{R})$$

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.³

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

When we tally the results we obtain the following score:

Total number of tokens	9
Unchanged	6
Substitutions	1
Insertions	2
Deletions	2
Simple Accuracy	0.44

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 accuracy metric may be negative (though it is never greater than 1, of course).

The simple 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 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').

³Note that the metric is symmetric.

(3)

$$\mathbf{GenerationAccuracy} = \left(1 - \frac{M + I' + D' + S}{R}\right)$$

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.⁴ Thus, the generation accuracy metric 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 as follows:

(4)

Metric	Simple Accuracy	Generation Accuracy
Total number of tokens	9	9
Unchanged	6	6
Substitutions	1	1
Insertions	2	1
Deletions	2	1
Moves	0	1
Score	0.44	0.67

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 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.

⁴This 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 accuracy would have 6 errors and hence be lower, but the generation accuracy would have 3 errors and hence be higher.

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. These two strings are evaluated by the simple accuracy and generation accuracy metrics discussed above. We then substitute the head word for the head-daughter strings in the reference and the generated strings, and the process is repeated for the next treelet, until only one word remains in the reference and generated strings. The sum of the errors for each of the string comparisons constitutes the errors for **Simple Tree Accuracy** and **Generation Tree Accuracy**.

Metric	Simple Accuracy	Generation 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
Score	0.44	0.56	0.33	0.67

3.3 Evaluation Results

We have performed various experiments employing different tree models. Here we tabulate the results of the four metrics on the output produced by using a baseline tree model and a supertag-based tree model (as described in Section 2) on the generation output.

- 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, 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 l_m with supertag s_m . Furthermore we use the supertag 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 1. As can be seen, the supertag-based model improves over the baseline LR model on all four baseline quantitative metrics.

Tree Model	Simple Accuracy	Generation Accuracy	Simple Tree Accuracy	Generation Tree Accuracy
Baseline LR Model	41.2%	56.2%	41.6%	63.2%
Supertag-based Model	58.9%	72.4%	65.7%	76.2%

Table 1: Performance results from the two tree models.

4 Qualitative Evaluation of the Quantitative Metrics

We have presented four metrics which we can compute automatically. However, as we have seen the metrics give rather different results. In order to verify which of the metrics should be used (or some combination thereof), we are currently performing evaluation experiments with human subjects. The goal of these experiments is to obtain one or two metrics which can be automatically computed, and which have been shown to significantly correlate with relevant human judgments.

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.)

Our main hypothesis is that the two tree-based metrics correlate better with both understandability and quality than the string-based metrics.

Our hypothesis is confirmed by some very initial results (based on two humans judging a total of 47 sentences). We have found four statistically significant pairwise correlations between understandability and quality on the one hand and the two tree-based metrics on the other hand, and no significant correlation involving the string-based metrics, except that the correlation between generation accuracy and quality approaches significance. In addition, the simple accuracy metric (but none of the other three) significantly correlates with the difference between quality and understandability (which is on average negative); this fact presumably reflects the fact that quality is affected directly by divergence from the word order in the WSJ.

For the final version of the paper, we will present the full results of the feedback experiment. We intend to use the human evaluation data in guiding us as needed in the design of further automatically derivable 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.

As in the case of machine translation, evaluation in generation is a complex issue. 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 one of the metrics can be 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.

References

- Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 1998. Automatic acquisition of hierarchical transduction models for machine translation. 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).
- Karen Kukich. 1983. *Knowledge-Based Report Generation: A Knowledge Engineering Approach to Natural Language Report Generation*. Ph.D. thesis, University of Pittsburgh.
- 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.
- 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.
- 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 <http://www.cis.upenn.edu/~xtag/tech-report/tech-report.html>, The Institute for Research in Cognitive Science, University of Pennsylvania.

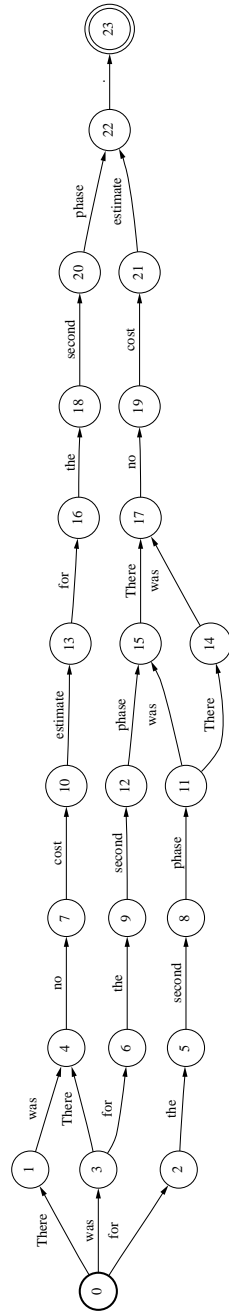


Figure 3: Word lattice for example sentence after Tree Chooser and Unraveler using the supertag-based model