Automatic Error Detection for Natural Language Generation

Michael Wayne Goodman[†] **Francis Bond**[‡]
[†] University of Washington
[‡] NICT Language Infrastructure Group, MASTAR Project
goodmami@u.washington.edu, bond@ieee.org

1 Introduction

One of the advantages of deep grammars, such as those based on HPSG, is that they can be used for generation as well as parsing. However, typically they are tested more thoroughly for parsing. In this paper we introduce a system that helps the grammar engineer efficiently bring the generation capabilities to the same level as the parsing. The system can quickly and accurately determine the characteristics of parsed and generated sentences from a deep grammar, and then nd the most salient grammar rules used in producing particular classes of sentences. For our purposes, we generally look at sentences that cannot generate or overgenerate, nd the problematic rules causing them to be that way, and attempt to x those rules.

Consider (1), the grammar was able to parse this sentence, but was unable to generate any new sentences from its semantics (Section 4.2 elaborates on this particular issue).

(1)	彼女	は	写真	写り	が	いい
	kanojyo	wa	shashin	utsuri	ga	ii
	she	TOP	picture	taking	NOM	good

She is good at taking pictures

The system is tested both on Jacy, an implemented HPSG grammar for Japanese and the ERG (English Resource Grammar). Often, by xing the problems we not only improve generation results, but also parsing coverage and accuracy.

1.1 Motivation

Bond et al. (2008) showed that by producing paraphrases of the English side of an aligned corpus, they could increase their corpus size and improve the performance of a statistical machine translation system. We want to increase the size of the Japanese side of the corpus for the same reasons. However, Jacy could not generate new sentences nearly as well as the ERG, and was thus unusable for the task. Table 1 shows the initial parsing and generation statistics for both the ERG and Jacy (further explanation of the table is in Section 2.1).

Improving generation would also greatly bene t to-Japanese machine translation tasks using the Jacy grammar. It has also been shown that by improving generation, one can generally increase parsing performance.

1.2 Resources

We use the Jacy (Siegel, 2000) Japanese grammar (hereafter “Jacy”) and the English Resource Grammar (ERG) (Flickinger, 2000) within the DELPH-IN[1] machinery (PET (Callmeier, 2000) for parsing and LKB (Copestake, 2002) for generating). These are rule-based systems based on the Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994) formalism for syntax, and Minimal Recursion Semantics (MRS) (Copestake et al., 2005) for semantic representation. These grammars can process a corpus of input sentences and output parsing and generation **profiles**, which include lists of processable items, their derivation trees, MRS, and surface form. We use the Tanaka Corpus (Tanaka, 2001) for both our English and Japanese input sentences. In this paper, we will refer to a corpus **item** as the collection of an input sentence, its parse information (parse trees, MRS), and its generation information (generated trees, MRS, surface forms).

Regarding the parsing and generation processes, we are paraphrasing sentences into the same language. This entails rst parsing the input sentence and obtaining the semantics from the best ranked parse(s), then using those semantics as input to generate new sentences. Therefore, we can only generate new sentences from sentences that could be parsed.

[1]Deep Linguistic Processing with HPSG Initiative – see http://www.delph-in.net for background information, including the list of current participants and pointers to available resources and documentation

	ERG		Jacy	
	abs	rel	abs	rel
Unparsable	13%		21%	
Ungeneratable	4%	4%	35%	44%
No original	12%	15%	34%	77%
Only original	34%	40%	1%	2%

Table 1: Initial general statistics

	ERG	Jacy
Lexemes differ	50%	90%
Tree differs	69%	75%
Rules differ	64%	56%
String differs	63%	95%
MRS differ	5%	10%

Table 2: Initial comparative statistics

2 Data Collection

Our error detection routine is a two-step process. First we analyze the parsing and generation pro les to determine the **characteristics** of each item. Characteristics include things such as lexical, syntactic, semantic, and surface similarity to the original parsed sentence. We will call a unique sequence of characteristics a **characteristic pattern**, abbreviated as **CP**. The second step in our error detection routine is to nd the most salient grammar rules for certain CPs. To do this step, we rst train a classi er with a simpli ed form of each sentence's syntactic structure as features and a string representation of each respective CP as labels. After training, we nd the rules that have the highest association to certain characteristics. In our tests, we found that the highest ranked rules for a class of problematic sentences indeed pointed to the most signi cant problems in those sentences.

2.1 Determining Characteristics

In Tables 1 and 2, we list nine different characteristics. The characteristics listed in Table 1 are general, process-related qualities, and those listed in Table 2 are comparative qualities between the parsed and generated sentences.

"Unparsable" means the grammar was unable to parse the input sentence, and "Ungenerable" means that it could not generate from the semantics created by a successful parse. "No original" means that the grammar could parse and generate from an input sentence, but could not generate the original parsed sentence. "Only original" means that the grammar could parse and generate a sentence, but could only generate the original sentence. This might be expected for very simple sentences.

Items where the "Lexemes differ" in the parsed and generated sentences will have a word inserted, deleted, or replaced with something different. "Rules differ" means that the (unordered) set of rules invoked when generating that sentence is different from the set invoked when parsing the original sentence. "Tree differs", on the other hand, occurs when the derivation tree (that is, the rules and their structure) differs from the parse's in some way. "String differs" occurs when the surface form of the parsed and generated sentences are different (ignoring some punctuation). Lastly, "MRS differ" means that the argument structure in the semantics are not equivalent. Because we are paraphrasing from the same semantic structure we should never have different semantics in the parsed and generated sentences.

2.2 Problematic Rule Ranking

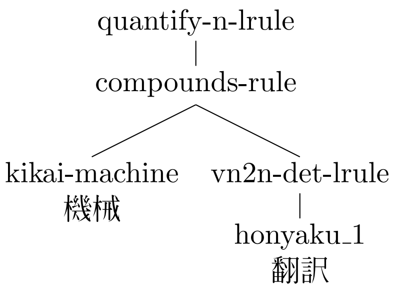
Once we have created a matrix of characteristic patterns for all input and output sentences, we train a maximum entropy-based classi er over the derivation tree and CPs. For each item, we extract a set of n-grams of nonbranching paths from the parse trees, creating separate paths for each branch in the original tree. These sets of paths are used as the features in the classi er. See Figure 1 for an example derivation tree and some paths extracted from it. We call these **rule paths**, abbreviated as **RP**. A string representation of the CPs from each item are then used as labels.

3 Results and Interpretation

While the ranking of problematic rules requires the characteristic patterns to be determined in order to run, both the ranked RPs and the CPs are useful individually to a grammar developer. We will describe how one can use each of these resources for grammar error detection.

3.1 Interpreting Characteristic Patterns

While some characteristics in a CP are mutually exclusive (e.g. "Unparsable" and "Ungenerable")



- quantify-n-lrule → compounds-rule → kikai-machine
- compounds-rule → kikai-machine → 機械
- quantify-n-lrule → compounds-rule → vn2n-det-lrule
- compounds-rule → vn2n-det-lrule → honyaku_1
- vn2n-det-lrule → honyaku_1 → 翻訳

Figure 1: Derivation tree for "機械翻訳" ("machine translation") and paths with length of 3

others often occur in parallel. Those that occur together are most likely not independent from each other. For instance, if a generated sentence has different lexemes from the original sentence, then it most likely will also have a different surface form.

Sometimes it can be useful to look at characteristics in combination. For example, every time an item has differing sets of rules it will invariably have differing trees as well. If, instead, the sets of rules are the same, but the tree structures differ, then there is a strong reason to believe that the clauses in the sentence were simply reordered. Similarly, if an item has identical sets of lexemes, sets of rules, and derivation trees, but has differing surface forms, then it is likely applying incorrect inflectional rules, since inflectional rules do not appear in the derivation tree.

3.2 Interpreting Problematic Rules

Our list of ranked rule paths is sorted on the score each RP received from the classi er. This is useful information, but it is incomplete. For example, the RP *vend-vend-rule → adj-te-t-lexeme-c-stem-infl-rule → nai-notyetend → "なく"* has a score of 0.34, and the RP *hf-complement-rule → head-specifier-rule → koto-pred → "こと"* has a (lower) score of 0.25, but the former only has 9 occurrences in the corpus, compared to the latter's 63. For the grammar developer looking to x bugs, the latter RP offers to yield more gain than the former, despite its higher score. Therefore, along with the score from the classi er, we also output the raw count of the items using that particular RP.

For a given list of ranked RPs, not all are unique. Since the RPs are n-grams, there are many overlapping or subset RPs, and often each variant will receive a similar score. To reduce the amount of cruft, we exclude RPs that are subsets or supersets of a previously listed RP. We do not exclude RPs that are merely shifted (such as 1-2-3 and 2-3-4) since there is less assurance that they do not actually describe a different problem.

For example, we saw that the highest ranking RP was *quantify-n-lrule → compounds-rule*, and it also occured many times in the corpus. This RP is a bigram that could be taken from the derivation tree in Figure 1. It turned out that the problem was indeed the interaction between the *quantify-n-lrule* and the *compounds-rule*. See Section 4.2 for more information on the solution to this problem.

4 Grammar Fixes

With the information provided by our system, we tried to x some of the most prevalent problems. The rst problem we xed was to restructure how the grammar treated commas and colons as topic markers. The second was to change the rule for noun quanti cation so compound nouns could generate. We modi ed the treatment of numeral classi ers so they compound (e.g. ５年契約 *go-nen keiyaku* "5-year contract"), stand alone as nouns they affect (e.g. 三匹が吠える *san-biki ga hoeru* "Three (animals) bark"), and generate properly in certain constructions (e.g. ３匹の犬 *san-biki no inu* "three dogs"). We removed some lexemes that caused spurious ambiguity, such as the hiragana versions of the verbs にる *niru* "to boil" and でる *deru* "to leave, to attend", as they would incorrectly be used in place of the *ni* and *de* case markers. From this (non-exhaustive) list of changes made to the grammar, we will describe the rst two in more detail.

The changes we made improved generation cov-

	before		after	
	abs	rel	abs	rel
Unparsable	21%		17%	
Ungeneratable	35%	44%	28%	34%
No original	34%	77%	41%	74%
Only original	1%	2%	1%	2%
Lexemes differ		90%		88%
Tree differs		75%		72%
Rules differ		56%		50%
String differs		95%		94%
MRS differ		10%		0.2%

Table 3: Jacy statistics, before and after

erage (over those items that could be parsed) by nearly 10% to 66%, while reducing overgeneration. See Table 3 for updated statistics from the improved grammar.

We also found some errors in the English grammar: e.g. the morphology is wrong for some gerunds: for the input *I showered before breakfast*, we generate *I showerred before breakfast*.

4.1 Overgenerating Topic Markers

In the initial version of Jacy, punctuation such as commas, colons, and equals could be treated as topic markers, which were a subtype of the *wa*-topic marker. This worked for parsing, but in generation we would overgenerate a comma, colon, or equals for every instance of a *wa*-topic marker as in (2). We solved this by making the topic-marker hierarchy more informative: comma, colon, equals, and *wa* were specialized to daughter types of a "topic" parent type. By making this change, we improved generation accuracy by eliminating many undesirable results. Parsing and generation coverage were not affected.

(2) a. その 計画 は 具体 化 し て き た
The project is taking shape
b. その 計画 、具体 化 し て き た
c. その 計画 ： 具体 化 し て き た
d. その 計画 ＝ 具体 化 し て き た

4.2 Ungenerable Compound Nouns

Compound nouns (like the one in Figure 1) caused almost every sentence in which they occurred to be ungenerable. The only sentences that could generate were those with explicit quanti ers, such as この *kono* "this". The problem was in the quanti cation of bare-nouns after compounding. The quanti cation rule was a lexical-rule and the generator could not apply it after a phrase-rule (such as the *compounds-rule*). Thus, we changed the quanti cation rule to be a phrase-rule, allowing compounds to generate. This resulted in a large boost in generation coverage. Fixing this bug led to changes in other areas, such as numeral classi ers. The extra changes not only resulted in greater generation coverage, but also increased parsing coverage by about 1.5%.

5 Conclusion

We have introduced a system that not only analyzes problems in an implemented HPSG grammar, but also nds and ranks the possible sources of those problems. This tool can greatly reduce the amount of time a grammar developer would spend nding bugs, and helps them make informed decisions about which bugs are best to x. Using our system, we were able to improve Jacy's generation coverage by 10% with only two weeks of grammar development.

References

Francis Bond, Eric Nichols, Darren Scott Appling, and Michael Paul. 2008. Improving statistical machine translation by paraphrasing the training data. In *International Workshop on Spoken Language Translation*, pages 150–157. Honolulu.

Ulrich Callmeier. 2000. PET - a platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6(1):99–108.

Ann Copestake. 2002. *Implementing Typed Feature Structure Grammars.* CSLI Publications.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal Recursion Semantics. An introduction. *Research on Language and Computation*, 3(4):281–332.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28. (Special Issue on Efficient Processing with HPSG).

Carl Pollard and Ivan A. Sag. 1994. *Head Driven Phrase Structure Grammar.* University of Chicago Press, Chicago.

Melanie Siegel. 2000. HPSG analysis of Japanese. In Wolfgang Wahlster, editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 265 – 280. Springer, Berlin, Germany.

Yasuhito Tanaka. 2001. Compilation of a multilingual parallel corpus. In *Proceedings of PACLING 2001*, pages 265–268. Kyushu. (`http://www.colips.org/afnlp/archives/pacling2001/pdf/tanaka.pdf`).