Probabilistic Generation of Weather Forecast Texts

Anja Belz
Natural Language Technology Group
University of Brighton, UK
a.s.belz@brighton.ac.uk

Abstract

This paper reports experiments in which *p*CRU — a generation framework that combines probabilistic generation methodology with a comprehensive model of the generation space — is used to semi-automatically create several versions of a weather forecast text generator. The generators are evaluated in terms of output quality, development time and computational efficiency against (i) human forecasters, (ii) a traditional handcrafted pipelined NLG system, and (iii) a HALOGEN-style statistical generator. The most striking result is that despite acquiring all decision-making abilities automatically, the best *p*CRU generators receive higher scores from human judges than forecasts written by experts.

1 Introduction and background

Over the last decade, there has been a lot of interest in statistical techniques among researchers in natural language generation (NLG), a field that was largely unaffected by the statistical revolution in NLP that started in the 1980s. Since Langkilde and Knight's influential work on statistical surface realisation (Knight and Langkilde, 1998), a number of statistical and corpus-based methods have been reported. However, this interest does not appear to have translated into practice: of the 30 implemented systems and modules with development starting in or after 2000 that are listed on a key NLG website[1], only five have any statistical component at all (another six involve techniques that are in some way corpus-based). The likely reasons for this lack of take-up are that (i) many existing statistical NLG techniques are inherently expensive, requiring the set of alternatives to be generated in full before the statistical model is applied to select the most likely; and (ii) statistical NLG techniques have not been shown to produce outputs of high enough quality.

There has also been a rethinking of the traditional modular NLG architecture (Reiter, 1994). Some research has moved towards a more comprehensive view, e.g. construing the generation task as a single constraint satisfaction problem. Precursors to current approaches were Hovy's PAULINE which kept track of the satisfaction status of global 'rhetorical goals' (Hovy, 1988), and Power et al.'s ICONOCLAST which allowed users to fine-tune different combinations of global constraints (Power, 2000). In recent comprehensive approaches, the focus is on automatic adaptability, e.g. automatically determining degrees of constraint violability on the basis of corpus frequencies. Examples include Langkilde's (2005) general approach to generation and parsing based on constraint optimisation, and Marciniak and Strube's (2005) integrated, globally optimisable network of classifiers and constraints.

Both probabilistic and recent comprehensive trends have developed at least in part to address two interrelated issues in NLG: the considerable amount

[1]Bateman and Zock's list of NLG systems, `http://www.fb10.uni-bremen.de/anglistik/langpro/NLG-table/`, 20/01/2006.

of time and expense involved in building new systems, and the almost complete lack in the field of reusable systems and modules. Both trends have the potential to improve on development time and reusability, but have drawbacks.

Existing statistical NLG (i) uses corpus statistics to inform heuristic decisions in what is otherwise symbolic generation (Varges and Mellish, 2001; White, 2004; Paiva and Evans, 2005); (ii) applies *n*-gram models to select the overall most likely realisation after generation (HALOGEN family); or (iii) reuses an existing parsing grammar or treebank for surface realisation (Velldal et al., 2004; Cahill and van Genabith, 2006). N-gram models are not linguistically informed, (i) and (iii) come with a substantial manual overhead, and (ii) overgenerates vastly and has a high computational cost (see also Section 3).

Existing comprehensive approaches tend to incur a manual overhead (finetuning in ICONOCLAST, corpus annotation in Langkilde and Marciniak & Strube). Handling violability of soft constraints is problematic, and converting corpus-derived probabilities into costs associated with constraints (Langkilde, Marciniak & Strube) turns straightforward statistics into an *ad hoc* search heuristic. Older approaches are not globally optimisable (PAULINE) or involve exhaustive search (ICONOCLAST).

The *p*CRU language generation framework combines a probabilistic generation methodology with a comprehensive model of the generation space, where probabilistic choice informs generation as it goes along, instead of after all alternatives have been generated. *p*CRU uses existing techniques (Belz, 2005), but extends these substantially. This paper describes the *p*CRU framework and reports experiments designed to rigorously test *p*CRU in practice and to determine whether improvements in development time and reusability can be achieved without sacrificing quality of outputs.

2 *p*CRU language generation

*p*CRU (Belz, 2006) is a probabilistic language generation framework that was developed with the aim of providing the formal underpinnings for creating NLG systems that are driven by comprehensive probabilistic models of the entire generation space (including deep generation). NLG systems tend to be composed of generation rules that apply transformations to representations (performing different tasks in different modules). The basic idea in *p*CRU is that as long as the generation rules are all of the form $relation(arg_1, ...arg_n) \rightarrow relation_1(arg_1, ...arg_p) ... relation_m(arg_1, ...arg_q)$, $m \geq 1, n, p, q \geq 0$, then the set of all generation rules can be seen as defining a context-free language and a single probabilistic model can be estimated from raw or annotated text to guide generation processes.

*p*CRU uses straightforward context-free technology in combination with underspecification techniques, to encode a **base generator** as a set of expansion rules G composed of n-ary relations with variable and constant arguments (Section 2.1). In non-probabilistic mode, the output is the set of fully expanded (fully specified) forms that can be derived from the input. The *p*CRU (*p*robabilistic CRU) **decision-maker** is created by estimating a probability distribution over the base generator from an unannotated corpus of example texts. This distribution is used in one of several ways to drive generation processes, maximising the likelihood either of individual expansions or of entire generation processes (Section 2.2).

2.1 Specifying the range of alternatives

Using context-free representational underspecification, or CRU, (Belz, 2004), the generation space is encoded as (i) a set G of expansion rules composed of n-ary relations $relation(arg_1, ...arg_n)$ where the arg_i are constants or variables over constants; and (ii) argument and relation type hierarchies. Any sentential form licensed by G can be the input to the generation process which expands it under unifying variable substitution until no further expansion is possible. The output (in non-probabilistic mode) is the set of fully expanded forms (i.e. consisting only of terminals) that can be derived from the input.

The rules in G define the steps in which inputs can be incrementally specified from, say, content to semantic, syntactic and finally surface representations. G therefore defines specificity relations between all sentential forms, i.e. defines which representation is underspecified with respect to which other representations. The generation process is construed explicitly as the task of incrementally specifying one or more word strings.

Within the limits of context-freeness and atomicity of feature values, CRU is neutral with respect to actual linguistic knowledge representation formalisms used to encode generation spaces. The main motivation for a context-free formalism is the advantage of low computational cost, while the inclusion of arguments on (non)terminals permits keeping track of contextual features.

2.2 Selection among alternatives

The *p*CRU decision-making component is created by estimating a probability distribution over the set of expansion rules that encodes the generation space (the base generator), as follows:

1 *Convert corpus into multi-treebank:* determine for each sentence all (left-most) derivation trees licensed by the base generator's CRU rules, using maximal partial derivations if there is no complete derivation tree; annotate the (sub)strings in the sentence with the derivation trees, resulting in a set of *generation trees* for the sentence.

2 *Train base generator:* Obtain frequency counts for each individual generation rule from the multi-treebank, adding $1/n$ to the count for every rule, where n is the number of alternative derivation trees; convert counts into probability distributions over alternative rules, using add-1 smoothing and standard maximum likelihood estimation.

The resulting probability distribution is used in one of the following three ways to control generation. Of these, only the first requires the generation forest to be created in full, whereas both greedy modes prune the generation space to a single path:

1 *Viterbi generation:* do a Viterbi search of the generation forest for a given input, which maximises the joint likelihood of all decisions taken in the generation process. This selects the most likely generation process, but is considerably more expensive than the greedy modes.

2 *Greedy generation:* make the single most likely decision at each choice point (rule expansion) in a generation process. This is not guaranteed to result in the most likely generation process, but the computational cost is very low.

3 *Greedy roulette-wheel generation:* use a non-uniform random distribution proportional to the likelihoods of alternatives. E.g. if there are two alternative decisions D_1 and D_2, with the model giving $p(D_1) = 0.8$ and $p(D_2) = 0.2$, then the proportion of times the generator decides D_1 approaches 80% and D_2 20% in the limit.

2.3 The *p*CRU-1.0 generation package

The technology described in the two preceding sections has been implemented in the *p*CRU-1.0 software package. The user defines a generation space by creating a base generator composed of:

1. the set N of underspecified n-ary relations
2. the set W of fully specified n-ary relations
3. a set R of context-free generation rules $n \rightarrow \alpha$, $n \in N$, $\alpha \in (W \cup N)^*$
4. a typed feature hierarchy defining argument types and values

This base generator is then trained (as described above) on raw text corpora to provide a probability distribution over generation rules. Optionally, an *n*-gram language model can also be created from the same corpus. The generator is then run in one of the three modes above or one of the following:

1. *Random*: ignoring *p*CRU probabilities, randomly select generation rules.
2. *N-gram*: ignoring *p*CRU probabilities, generate set of alternatives and select the most likely according to the *n*-gram language model.

The random mode serves as a baseline for generation quality: a trained generator must be able to do better, otherwise all the work is done by the base generator (and none by the probabilities). The *n*-gram mode works exactly like HALOGEN-style generation: the generator generates all realisations that the rules allow and then picks one based on the *n*-gram model. This is a point of comparison with existing statistical NLG techniques and also serves as a baseline in terms of computational expense: a generator using *p*CRU probabilities should be able to produce realisations faster.

3 Building and evaluating *p*CRU wind forecast text generators

The automatic generation of weather forecasts is one of the success stories of NLP. The restrictiveness of the sublanguage has made the domain of

```
Oil1/Oil2/Oil3_FIELDS
05-10-00

05/06 SSW 18  22  27   3.0  4.8 SSW  2.59
05/09 S   16  20  25   2.7  4.3 SSW  2.39
05/12 S   14  17  21   2.5  4.0 SSW  2.29
05/15 S   14  17  21   2.3  3.7 SSW  2.28
...


FORECAST FOR:-
Oil1/Oil2/Oil3 FIELDS
...
2.FORECAST 06-24 GMT, THURSDAY, 05-Oct   2000
=====WARNINGS:   RISK THUNDERSTORM.  =======
WIND(KTS)   CONFIDENCE: HIGH
  10M:      SSW 16-20 GRADUALLY BACKING SSE
            THEN FALLING VARIABLE 04-08 BY
            LATE EVENING
...
```

Figure 1: Meteorological data file and wind forecast for 05-10-2000, a.m. (oil fields anonymised).

weather forecasting particularly attractive to NLG researchers, and a number of weather forecast generation systems have been created.

A recent example of weather forecast text generation is the SUMTIME project (Reiter et al., 2005) which developed a commercially used NLG system that generates marine weather forecasts for offshore oil rigs from numerical forecast data produced by weather simulation programs. The SUMTIME corpus is used in the experiments below.

3.1 Data

Each instance in the SUMTIME corpus consists of three numerical data files (the outputs of weather simulators) and the forecast file written by the forecaster on the basis of the data (Figure 1 shows an example). The experiments below focused on a.m. forecasts of wind characteristics. Content determination (deciding which meteorological data to include in a forecast) was carried out off-line.

The corpus consists of 2,123 instances (22,985 words) of which half are a.m. forecasts. This may not seem much, but considering the small number of vocabulary items and syntactic structures, the corpus provides extremely good coverage (an initial impression confirmed by the small differences between training and testing data results below).

3.2 The base generator

The base generator[2] was written semi-automatically in two steps. First, a simple chunker was run over the corpus to split wind statements into wind direction, wind speed, gust speed, gust statements, time expressions, verb phrases, pre-modifiers, and post-modifiers. Preterminal generation rules were automatically created from the resulting chunks. Then, higher-level rules which combine chunks into larger components, taking care of text structuring, aggregation and elision, were manually authored. The top-level generation rules interpret wind statements as sequences of independent units of information, ensuring a linear increase in complexity with increasing input length. Inputs encode meteorological data (as shown in Table 1), and were pre-processed to determine certain types of information, including whether a change in wind direction was clockwise or anti-clockwise, and whether change in wind speed was an increase or a decrease. The final generator takes as inputs number vectors of length 7 to 60, and generates up to 1.6×10^{31} alternative realisations for an input.

The job of the base generator is to describe the textual variety found in the corpus. It makes no decisions about when to prefer one variant over another.

3.3 Training

The corpus was divided at random into 90% training data and 10% testing data. The training set was multi-treebanked with the base generator and the multi-treebank then used to create the probability distribution for the base generator (as described in Section 2.2). A back-off 2-gram model with Good-Turing discounting and no lexical classes was also created from the training set, using the SRILM toolkit, (Stolcke, 2002). *p*CRU-1.0 was then run in all five modes to generate forecasts for the inputs in both training and test sets.

This procedure was repeated five times for hold-out cross-validation. The small amount of variation across the five repeats, and the small differences between results for training and test sets (Table 2) indicated that five repeats were sufficient.

3.4 Evaluation

3.4.1 Evaluation methods

The two automatic metrics used in the evaluations, NIST and BLEU have been shown to correlate highly with expert judgments (Pearson correlation coefficients 0.82 and 0.79 respectively) in this domain (Belz and Reiter, 2006).

[2]For a fragment of the rule set, see Belz (2006).

Input	`[[1,SSW,16,20,-,-,0600],[2,SSE,-,-,-,-,NOTIME],[3,VAR,04,08,-,-,2400]]`
Corpus	`SSW 16-20 GRADUALLY BACKING SSE THEN FALLING VARIABLE 4-8 BY LATE EVENING`
Reference 1	`SSW'LY 16-20 GRADUALLY BACKING SSE'LY THEN DECREASING VARIABLE 4-8 BY LATE EVENING`
Reference 2	`SSW 16-20 GRADUALLY BACKING SSE BY 1800 THEN FALLING VARIABLE 4-8 BY LATE EVENING`
SumTime-Hyb.	`SSW 16-20 GRADUALLY BACKING SSE THEN BECOMING VARIABLE 10 OR LESS BY MIDNIGHT`
*p*CRU-greedy	`SSW 16-20 BACKING SSE FOR A TIME THEN FALLING VARIABLE 4-8 BY LATE EVENING`
*p*CRU-roulette	`SSW 16-20 GRADUALLY BACKING SSE AND VARIABLE 4-8`
*p*CRU-viterbi	`SSW 16-20 BACKING SSE VARIABLE 4-8 LATER`
*p*CRU-2gram	`SSW 16-20 BACKING SSE VARIABLE 4-8 LATER`
*p*CRU-random	`SSW 16-20 AT FIRST FROM MIDDAY BECOMING SSE DURING THE AFTERNOON THEN VARIABLE 4-8`

Table 1: Forecast texts (for 05-10-2000) generated by each of the *p*CRU generators, the SumTime-Hybrid system and three experts. The corresponding input to the generators is shown in the first row.

BLEU (Papineni et al., 2002) is a precision metric that assesses the quality of a translation in terms of the proportion of its word *n*-grams ($n \leq 4$ has become standard) that it shares with several reference translations. BLEU also incorporates a 'brevity penalty' to counteract scores increasing as length decreases. BLEU scores range from 0 to 1.

The NIST metric (Doddington, 2002) is an adaptation of BLEU, but where BLEU gives equal weight to all *n*-grams, NIST gives more weight to less frequent (hence more informative) *n*-grams. There is evidence that NIST correlates better with human judgments than BLEU (Doddington, 2002; Belz and Reiter, 2006).

The results below include human scores from two separate experiments. The first was an experiment with 9 subjects experienced in reading marine forecasts (Belz and Reiter, 2006), the second is a new experiment with 14 similarly experienced subjects[3]. The main differences were that in Experiment 1, subjects rated on a scale from 0 to 5 and were asked for overall quality scores, whereas in Experiment 2, subjects rated on a 1–7 scale and were asked for language quality scores.

In comparing different *p*CRU modes, NIST and BLEU scores were computed against the test set part of the corpus which contains texts by five different authors. In the two human experiments, NIST and BLEU scores were computed against sets of multiple reference texts (2 for each date in Experiment 1, and 3 in Experiment 2) written by forecasters who had not contributed to the corpus. One-way ANOVAs with post-hoc Tukey HSD tests were used to analyse variance and statistical significance of all results.

Table 1 shows forecast texts generated by each of the systems included in the evaluations reported below, together with the corresponding input and three texts created by humans for the same data.

[3]Belz and Reiter, in preparation.

		NIST-5	BLEU-4
T	*p*CRU-greedy	8.208 (0.033)	0.647 (0.002)
R	*p*CRU-roulette	7.035 (0.138)	0.496 (0.010)
A	*p*CRU-2gram	6.734 (0.086)	0.523 (0.008)
I	*p*CRU-viterbi	6.643 (0.023)	0.524 (0.002)
N	*p*CRU-random	4.799 (0.036)	0.296 (0.002)
	*p*CRU-greedy	6.927 (0.131)	0.636 (0.016)
T	*p*CRU-roulette	6.193 (0.121)	0.496 (0.022)
E	*p*CRU-2gram	5.663 (0.185)	0.514 (0.019)
S	*p*CRU-viterbi	5.650 (0.161)	0.519 (0.021)
T	*p*CRU-random	4.535 (0.078)	0.313 (0.005)

Table 2: NIST-5 and BLEU-4 scores for training and test sets (average variation from the mean).

3.4.2 Comparing different generation modes

Table 2 shows results for the five different *p*CRU generation modes, for training sets (top) and test sets (bottom), in terms of NIST-5 and BLEU-4 scores averaged over the five runs of the hold-out validation, with average mean deviation figures across the runs shown in brackets.

The Tukey Test produced the following results for the differences between means in Table 2. For the training set, results are the same for NIST and BLEU scores: all differences are significant at $P < 0.01$, except for the differences in scores for *p*CRU-2gram and *p*CRU-viterbi. For the test set and NIST, again all differences are significant at $P < 0.01$, except for *p*CRU-2gram vs. *p*CRU-viterbi. For the test set and BLEU, three differences are non-significant: *p*CRU-2gram vs. *p*CRU-viterbi, *p*CRU-2gram vs. *p*CRU-

	Experiment 1	Experiment 2
SUMTIME-Hyb.	3.82 (1)	4.61 (2)
*p*CRU-greedy	3.59 (2)	4.79 (1)
*p*CRU-roulette	3.22 (3)	4.54 (3)

Table 3: Scores for handcrafted system and two best *p*CRU-systems from two human experiments.

roulette, and *p*CRU-viterbi vs. *p*CRU-roulette.

NIST-5 depends on test set size, and is necessarily lower for the (smaller) test set, but the BLEU-4 scores indicate that performance was slightly worse on test sets. The deviation figures show that variation was also higher on the test sets.

The clearest result is that *p*CRU-greedy is ranked highest, and *p*CRU-random lowest, by considerable margins. *p*CRU-roulette is ranked second by NIST-5 and fourth by BLEU-4. *p*CRU-2gram and *p*CRU-viterbi are virtually indistinguishable.

Experts in both human experiments agreed with the NIST-5 rankings of the modes exactly.

3.4.3 Text quality against handcrafted system

The *p*CRU modes were also evaluated against the SUMTIME-Hybrid system (running in 'hybrid' mode, taking inputs as in Table 1). Table 3 shows averaged evaluation scores by subjects in the two independent experiments described above. There were altogether 6 and 7 systems evaluated in these experiments, respectively, and the differences between the scores shown here were not significant when subjected to the Tukey Test, meaning that both experiments failed to show that experts can tell the difference in the language quality of the texts generated by the handcrafted SUMTIME-Hybrid system and the two best *p*CRU-greedy systems.

3.4.4 Text quality against human forecasters

In the first experiment, the human evaluators gave an average score of 3.59 to *p*CRU-greedy, 3.22 to the corpus texts, and 3.03 to another (human) forecaster. In Experiment 2, the average human scores were 4.79 for *p*CRU-greedy, and 4.50 for the corpus texts. Although in each experiment separately, statistical significance could not be shown for the differences between these means, in combination the scores provide evidence that the evaluators thought *p*CRU-greedy better than the human-written texts.

3.4.5 Computing time

The following table shows average number of seconds taken to generate one forecast, averaged over the five cross-validation runs (mean variation figures across the runs in brackets):

	Training sets	Test sets
*p*CRU-greedy:	1.65s $(= 0.02)$	1.58s (< 0.04)
*p*CRU-roulette:	1.61s (< 0.02)	1.58s (< 0.05)
*p*CRU-viterbi:	1.74s (< 0.02)	1.70s $(= 0.04)$
*p*CRU-2gram:	2.83s (< 0.02)	2.78s (< 0.09)

Forecasts for the test sets were generated somewhat faster than for the training sets in all modes. Variation was greater for test sets. Differences between *p*CRU-greedy and *p*CRU-roulette are very small, but *p*CRU-viterbi took $1/10$ of a second longer, and *p*CRU-2gram took more than 1 second longer to generate the average forecast[4].

3.4.6 Brevity bias

N-gram models have a built-in bias in favour of shorter strings, because they calculate the likelihood of a string of words as the joint probability of the words, or, more precisely, as the product of the probabilities of each word given the $n - 1$ preceding words. The likelihood of any string will therefore generally be lower than that of any of its substrings.

Using a smaller data set for which all systems had outputs, the average number of words in the forecasts generated by the different systems was:

*p*CRU-random:	19.43
SUMTIME-Hybrid:	12.39
*p*CRU-greedy:	11.51
Corpus:	*11.28*
*p*CRU-roulette:	10.48
*p*CRU-2gram:	7.66
*p*CRU-viterbi:	7.54

*p*CRU-random has no preference for shorter strings, its average string length is almost twice that of the other *p*CRU-generators. The 2-gram generator prefers shorter strings, while the Viterbi generator prefers shorter generation processes, and these preferences result in the shortest texts. The poor evaluation results above for the n-gram and Viterbi generators indicate that this brevity bias can be harm-

[4]The Viterbi and the 2-gram generator were implemented identically, except for the *n*-gram model look-up.

ful in NLG. The remaining generators achieve good matches to the average forecast length in the corpus.

3.4.7 Development time

The most time-consuming part of NLG system development is not encoding the range of alternatives, but the decision-making capabilities that enable selection among them. In SUMTIME (Section 3), these were the result of corpus analysis and consultation with writers and readers of marine forecasts. In the *p*CRU wind forecast generators, the decision-making capabilities are acquired automatically, no expert knowledge or corpus annotation is used.

The SUMTIME team estimate[5] that very approximately 12 person months went directly into developing the SUMTIME microplanner and realiser (the components functionally analogous to the *p*CRU-generators), and 24 on generic activities such as expert consultation, which also benefited the microplanner/realiser. The *p*CRU wind forecasters were built in less than a month, including familiarisation with the corpus, building the chunker and creating the generation rules themselves. However, the SUMTIME system also generates wave forecasts and appropriate layout and canned text. A generous estimate is that it would take another two person months to equip the *p*CRU forecaster with these capabilities.

This is not to say that the two research efforts resulted in exactly the same thing. It is clear that forecast readers prefer the SUMTIME system, but the point is that it did come with a substantial price tag attached. The *p*CRU approach allows control over the trade-off between cost and quality.

4 Discussion

The main contributions of the research described in this paper are: (i) a generation methodology that improves substantially on development time and reusability compared to traditional hand-crafted systems; (ii) techniques for training linguistically informed decision-making components for probabilistic NLG from raw corpora; and (iii) results that show that probabilistic NLG can produce high-quality text. Results also show that (i) a preference for shorter realisations can be harmful in NLG; and that (ii) linguistically literate, probabilistic NLG can outperform HALOGEN-style shallow statistical methods, in terms of quality and efficiency.

[5]Personal communication with E. Reiter and S. Sripada.

An interesting question concerns the contribution of the manually built component (the base generator) to the quality of the outputs. The random mode serves as an absolute baseline in this respect: it indicates how well a particular base generator performs on its own. However, different base generators have different effects on the generation modes. The base generator that was used in previous experiments (Belz, 2005) encoded a less structured generation space and the set of concepts it used were less fine-grained (e.g. it did not distinguish between an increase and a decrease in wind speed, considering both simply a change), and therefore it lacked some information necessary for deriving conditional probabilities for lexical choice (e.g. *freshening* vs. *easing*). As predicted (Belz, 2005, p. 21), improvements to the base generator made little difference to the results for *p*CRU-2gram (up from BLEU 0.45 to 0.5), but greatly improved the performance of the greedy mode (up from 0.43 to 0.64).

A basic question for statistical NLG is whether surface string likelihoods are enough to resolve remaining non-determinism in generators, or whether likelihoods at the more abstract level of generation rules are needed. The former always prefers the most frequent variant regardless of context, whereas in the latter probabilities can attach to linguistic objects and be conditioned on contextual features (e.g. one useful feature in the forecast text generators encoded whether a rule was being applied at the beginning of a text). The results reported in this paper provide evidence that probabilistic generation can be more powerful than n-gram based post-selection.

5 Conclusions

The *p*CRU approach to generation makes it possible to combine the potential accuracy and subtlety of symbolic generation rules with detailed linguistic features on the one hand, and the robustness and handle on nondeterminism provided by probabilities associated with these rules, on the other. The evaluation results for the *p*CRU generators show that outputs of high quality can be produced with this approach, that it can speed up development and improve reusability of systems, and that in some modes it is more efficient and less brevity-biased than existing HALOGEN-style *n*-gram techniques.

The current situation in NLG recalls NLU in the late 1980s, when symbolic and statistical NLP were separate research paradigms, a situation memorably caricatured by Gazdar (1996), before rapidly moving towards a paradigm merger in the early 1990s. A similar development is currently underway in MT where — after several years of statistical MT dominating the field — researchers are now beginning to bring linguistic knowledge into statistical techniques (Charniak et al., 2003; Huang et al., 2006), and this trend looks set to continue. The lesson from NLU and MT appears to be that higher quality results when the symbolic and statistical paradigms join forces. The research reported in this paper is intended to be a first step in this direction for NLG.

Acknowledgments

This research was in part supported under UK EPSRC Grant GR/S24480/01. Many thanks to the anonymous reviewers for very helpful comments.

References

A. Belz and E. Reiter. 2006. Comparing automatic and human evaluation of NLG systems. In *Proc. EACL'06*, pages 313–320.

A. Belz. 2004. Context-free representational underspecification for NLG. Technical Report ITRI-04-08, University of Brighton.

A. Belz. 2005. Statistical generation: Three methods compared and evaluated. In *Proc. of ENLG'05*, pages 15–23.

A. Belz. 2006. *p*CRU: Probabilistic generation using representational underspecification. Technical Report NLTG-06-01, University of Brighton.

A. Cahill and J. van Genabith. 2006. Robust PCFG-based generation using automatically acquired LFG approximations. In *Proc. ACL'06*, pages 1033–44.

E. Charniak, K. Knight, and K. Yamada. 2003. Syntax-based language models for machine translation. In *Proc. MT Summit IX*.

G. Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the ARPA Workshop on Human Language Technology*.

G. Gazdar. 1996. Paradigm merger in NLP. In Robin Milner and Ian Wand, editors, *Computing Tomorrow: Future Research Directions in Computer Science*, pages 88–109. Cambridge University Press.

E. Hovy. 1988. *Generating Natural Language under Pragmatic Constraints*. Lawrence Erlbaum.

L. Huang, K. Knight, and A. Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. AMTA*, pages 66–73.

K. Knight and I. Langkilde. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of COLING-ACL'98*, pages 704–710.

I. Langkilde. 2005. An exploratory application of constraint optimization in Mozart to probabilistic natural language processing. In *Proceedings of CSLP'05*, volume 3438 of *LNAI*. Springer-Verlag.

T. Marciniak and M. Strube. 2005. Using an annotated corpus as a knowledge source for language generation. In *Proceedings of UCNLG'05*, pages 19–24.

D. S. Paiva and R. Evans. 2005. Empirically-based control of natural language generation. In *Proceedings ACL'05*.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proc. ACL '02*, pages 311–318.

R. Power. 2000. Planning texts by constraint satisfaction. In *Proceedings of COLING'00*.

E. Reiter, S. Sripada, J. Hunter, and J. Yu. 2005. Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167:137–169.

E. Reiter. 1994. Has a consensus NL generation architecture appeared and is it psycholinguistically plausible? In *Proceedings of INLG'94*, pages 163–170.

A. Stolcke. 2002. SRILM: An extensible language modeling toolkit. In *Proceedings of ICSLP'02*, pages 901–904,.

S. Varges and C. Mellish. 2001. Instance-based NLG. In *Proc. of NAACL'01*, pages 1–8.

E. Velldal, S. Oepen, and D. Flickinger. 2004. Paraphrasing treebanks for stochastic realization ranking. In *Proc. of TLT'04*.

M. White. 2004. Reining in CCG chart realization. In *Proceedings INLG'04*, volume 3123 of *LNAI*, pages 182–191. Springer.