

# High-quality Probabilistic Generation of Language

Anja Belz

Natural Language Technology Group

University of Brighton, UK

a.s.belz@brighton.ac.uk

## Abstract

Two important recent trends in NLG are (i) probabilistic techniques and (ii) comprehensive approaches that move away from traditional strictly modular and sequential models. This paper reports experiments in which *p*CRU — a generation framework that combines probabilistic generation methodology with a comprehensive model of the generation space — was used to semi-automatically create five different versions of a weather forecast 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 produce outputs of high enough quality to be judged better by 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<sup>1</sup>, only five have any statistical component at all (another six involving 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 simply have not been shown to produce outputs of high enough quality.

At the same time, there has 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 constraint satisfaction problem. Precursors to current approaches were Hovy's PAULINE which kept track of the satisfaction status of a set of global 'rhetorical goals' (Hovy, 1988), and Power et al.'s ICONOCLAST which allowed users to fine-tune different combinations of constraints (Power, 2000). In more recent comprehensive approaches, the focus is on automatic adaptability, e.g. automatically determining degrees of violability of constraints on the basis of corpus frequencies. Examples include Langkilde's (2005) general approach to both generation and parsing based on constraint optimisation, and Marciniak and Strube's (2005) integrated and 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 of time and expense involved in building new systems, and the almost complete lack in

---

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

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 methods either use statistics derived from a corpus to inform heuristic decisions during what is otherwise symbolic generation (Varges and Mellish, 2001; White, 2004; Paiva and Evans, 2005), or they use  $n$ -gram models to select the overall most likely realisation after generation (HALOGEN family). The automatic adaptability of the former is somewhat limited, and the latter overgenerates vastly, has a high computational cost (see also Section 3), and is not linguistically informed.

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

The  $p$ CRU language generation framework was developed as an efficient, linguistically informed, statistically principled alternative. It combines a probabilistic generation methodology with a comprehensive model of the generation space, where probabilistic choice informs generation as it goes along, instead of generating all alternatives and then selecting probabilistically.  $p$ CRU uses existing techniques (Belz, 2005), but extends these substantially. This paper describes the  $p$ CRU framework and reports new 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 (in the EPSRC project Cogent) 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 representa-

tions (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  $relation(arg_1, \dots, arg_n) \rightarrow relation_1(arg_1, \dots, arg_p) \dots relation_m(arg_1, \dots, arg_q)$ ,  $m \geq 1, n, p, q \geq 0$ , 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 (context-free representational underspecification, CRU, (Belz, 2004)), to encode a **base generator** as (i) a set  $G$  of expansion rules (of the form above) composed of  $n$ -ary relations  $relation(arg_1, \dots, arg_n)$  where the  $arg_i$  are constants or variables over constants; and (ii) argument and relation type hierarchies. During generation, inputs are expanded under unifying variable substitution until no further expansion is possible. 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 (probabilistic CRU) **decision-maker** is created by estimating a probability distribution over the base generator from an unannotated corpus of example texts, in two steps. First, the corpus is converted to a **multi-treebank**. For each sentence, all (left-most) derivation trees licensed by  $G$  are determined and added to the corpus. In the second step, frequency counts are determined for each individual generation rule from the multi-treebank. The counts are converted into a probability distribution over  $G$ , using smoothing and standard maximum likelihood estimation. 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.

### 2.1 Specifying the range of alternatives

Using context-free representational underspecification, or CRU, (Belz, 2004), the generation space is encoded as a set of expansion rules  $G$  composed of  $n$ -ary relations with variable and constant arguments. Any sentential form licensed by  $G$  can be the input to the generation process which expands the input under unifying variable substitution until no further expansion is possible. The output is the set of fully expanded (fully specified) forms that can be derived from the input.  $G$  places all input, intermediate and output representations

in specificity relations, or viewed the other way around, 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.

## 2.2 Selection among alternatives

The *p*CRU (probabilistic 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 three ways to drive a generation process:

1. *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.
2. *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 does select the most likely generation process, but is considerably more expensive.
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  $D1$  and  $D2$ , with the model giving  $p(D1) = 0.8$  and  $p(D2) = 0.2$ , then the generator decides  $D1$  approximately 80% of the time, and  $D2$  20%.

## 2.3 The *p*CRU-1.0 generation package

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

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 (in the way described in the previous section) 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 described in the previous section or one of the following two:

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 an absolute 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 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 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 weather forecasting particularly attractive to

```

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
05/18 SSE 12 15 18 2.4 3.8 SSW 2.38
05/21 SSE 10 12 15 2.4 3.8 SSW 2.48
06/00 VAR 6 7 8 2.4 3.8 SSW 2.48
...

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).

NLG researchers, and a number of weather forecast generation systems have been created, of which the two best known are perhaps METEO (Isabelle, 1984) and FoG (Goldberg et al., 1994).

A recent example of weather forecast 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 corpus developed for SUMTIME (Sripada et al., 2002) is used in the experiments below.

### 3.1 Data

Each instance in the SUMTIME corpus consists of three numerical data files (produced by three different weather simulators) and the weather forecast file written by the forecaster on the evidence of the data files. The experiments below focused on the part of the forecasts that predicts wind characteristics for the next 15 hours. Content determination (here, the task of deciding which meteorological data to include in a forecast) was carried out off-line. Table 1 includes an example content representation that formed the input to generators.

The corpus consisted of 2,123 instances, corresponding to a total of 22,985 words. This may not sound like 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 was written semi-automatically by running a set of simple chunking rules over the corpus that split wind statements into wind direction, wind speed, gust speed, gust statements, time expressions, verb phrases, pre-modifiers, and post-modifiers.

The manual part was to write the chunking rules, preterminal (lexical) rules, and higher-level rules that combine chunks into larger components, taking care of text structuring, aggregation and elision. The top-level generation rules interpreted wind statements as sequences of independent units of information, ensuring a linear increase in complexity with increasing input length. Inputs 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  $1.6 \times 10^{31}$  alternative realisations.

### 3.3 Training

The corpus was divided at random into training and testing data at a ratio of 9:1. 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 the training and the 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<sup>2</sup> and BLEU<sup>3</sup>, 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).

<sup>2</sup>[http://cio.nist.gov/esd/emaildir/lists/mt\\_list/bin00000.bin](http://cio.nist.gov/esd/emaildir/lists/mt_list/bin00000.bin)

<sup>3</sup><ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl>

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	SSW 16-20 GRADUALLY BACKING SSE THEN BECOMING VARIABLE 10 OR LESS BY MIDNIGHT
<i>p</i> CRU-greedy	SSW 16-20 BACKING SSE FOR A TIME THEN FALLING VARIABLE 4-8 BY LATE EVENING
<i>p</i> CRU-roulette	SSW 16-20 GRADUALLY BACKING SSE AND VARIABLE 4-8
<i>p</i> CRU-viterbi	SSW 16-20 BACKING SSE VARIABLE 4-8 LATER
<i>p</i> CRU-2gram	SSW 16-20 BACKING SSE VARIABLE 4-8 LATER
<i>p</i> CRU-random	SSW 16-20 AT FIRST FROM MIDDAY BECOMING SSE DURING THE AFTERNOON THEN VARIABLE 4-8

Table 1: Example input with corresponding forecast texts (for 05-10-2000).

BLEU (Papineni et al., 2002) is a precision metric (with brevity penalty) that assesses the quality of a translation in terms of the proportion of its word  $n$ -grams ( $n = 4$  has become standard) that it shares with several reference translations. 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 importance to less frequent (hence more informative)  $n$ -grams. Research has shown NIST to correlate with human judgments more highly than BLEU (Doddington, 2002; Riezler and Maxwell III, 2005; Belz and Reiter, 2006). In the experiments reported here, NIST and BLEU scores were computed against two sets of reference texts written by forecasters who had not contributed to the corpus.

The human evaluation results are from experiments with 9 experts (people with experience reading marine forecasts) and 21 non-experts (Belz and Reiter, 2006). Subjects did not have a background in NLP, and were native speakers of English. They were shown forecast texts from all the generators and from the corpus, and asked to score them on a scale of 0 to 5, for readability, clarity and general appropriateness. Experts were additionally shown the weather data corresponding to the forecast texts.

The statistical significance test was approximate randomisation (AR), as recommended by Riezler and Maxwell III (2005).

Table 1 shows example outputs by the humans and systems included in the evaluations below.

### 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-

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

out validation, with average mean deviation figures across the runs shown in brackets.

Pair-wise AR tests between results in Table 2 showed all differences to be significant with  $p < 0.05$  (which has become the accepted significance threshold in NLP), except for the differences in scores for *p*CRU-2gram and *p*CRU-viterbi.

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 and non-experts in the human-based evaluations agreed with the NIST-5 rankings exactly (see Table 4 below)<sup>4</sup>.

<sup>4</sup>*p*CRU-viterbi was not included in the human evaluations because it produces the same output as *p*CRU-2gram nearly all the time (and expert time was at a premium)

	Experts	Non-exp	NIST-5
SUMTIME-H.	0.76 (1)	0.77 (1)	5.98 (2)
<i>p</i> CRU-greedy	0.71 (2)	0.68 (3)	6.54 (1)
<i>p</i> CRU-roulette	0.62 (3)	0.71 (2)	5.83 (3)

Table 3: Evaluation scores for set of 18 forecasts used in expert evaluation, NIST-5 scores against two sets of reference texts.

### 3.4.3 Text quality against handcrafted system

The *p*CRU modes were also evaluated against the SUMTIME system (running in ‘hybrid’ mode with off-line content determination). Table 3 shows human evaluation scores and NIST-5 scores for SUMTIME-Hybrid and the two best *p*CRU generators for the 18 forecasts that were used in the expert evaluations.

On this small test set, the main result is that both experts and non-experts score SUMTIME-Hybrid the highest, whereas the metric scores *p*CRU-greedy highest.

The main difference between SUMTIME-Hybrid and the *p*CRU generators is in their decision-making mechanisms. The *p*CRU generators only had the frequencies of occurrence and co-occurrence of words and phrases in the corpus to go on and so their choices always reflect corpus frequency in some way. The SUMTIME generator deliberately deviates from highest corpus frequency where user studies showed a user preference for other alternatives. The human scores confirm that this improved forecasts from the readers’ perspective, whereas the metric necessarily prefers the *p*CRU-greedy texts which are more similar to the reference texts in terms of string similarity.

### 3.4.4 Text quality against human forecasters

Table 4 shows the full set of human evaluation scores (experts on 18 forecasts, non-experts on 21) for all *p*CRU modes, the corpus texts and one of the reference text sets (‘Human-A’). Variation arising from individual evaluator and from date is shown in brackets as average mean deviation. Experts and non-experts rank the humans and systems identically except that the experts rank *p*CRU-greedy above the corpus texts, and Human-A above *p*CRU-2gram.

The variation figures give an indication of statistical significance, and reveal that the differences between the scores of systems with adjacent ranks may not be large enough to be significant. Over-

	Experts (variation from subject, date)	Non-experts (variation from subject, date)
<i>p</i> CRU-greedy	0.71 (0.14,0.13)	0.7 (0.10,0.12)
Corpus	0.64 (0.20,0.13)	0.72 (0.13,0.15)
<i>p</i> CRU-roulette	0.62 (0.09,0.17)	0.69 (0.12,0.17)
Human-A	0.60 (0.12,0.14)	0.61 (0.14,0.14)
<i>p</i> CRU-2gram	0.53 (0.13,0.17)	0.65 (0.12,0.21)
<i>p</i> CRU-random	0.48 (0.17,0.16)	0.5 (0.16,0.17)

Table 4: Scores by experts and non-experts.

all, the scores can be taken to show that the experts judged (i) *p*CRU-greedy at least as good as the corpus texts, and better than Human-A, and (ii) *p*CRU-roulette at least as good as Human-A.

### 3.4.5 Computing time

The following table shows 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
<i>p</i> CRU-greedy:	1.65s (= 0.02)	1.58s (< 0.04)
<i>p</i> CRU-roulette:	1.61s (< 0.02)	1.58s (< 0.05)
<i>p</i> CRU-Viterbi:	1.74s (< 0.02)	1.70s (= 0.04)
<i>p</i> 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. Given the amount of variation, differences between *p*CRU-greedy and *p*CRU-roulette are not significant, 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<sup>5</sup>.

### 3.4.6 Brevity bias

*N*-gram models have a built-in bias in favour of shorter strings. This is 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 as follows:

<sup>5</sup>The Viterbi and the 2-gram generator were implemented identically, except for the *n*-gram model look-up.

<i>p</i> CRU-random:	19.43
SUMTIME	12.39
<i>p</i> CRU-greedy:	11.51
<i>Corpus</i> :	11.28
<i>p</i> CRU-roulette:	10.48
<i>p</i> CRU-2gram:	7.66
<i>p</i> CRU-Viterbi:	7.54

The random *p*CRU-generator has no preference for shorter strings at all, and has an average string length 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 2-gram and Viterbi generators show that this brevity bias is harmful in NLG. The remaining generators achieve good matches to the average forecast length in the corpus.

### 3.4.7 Development time

As argued above (Section 2), what takes most time in developing NLG systems is not encoding the range of alternatives, but the decision-making capabilities that enable selection among them. In SUMTIME, 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 were acquired entirely automatically: no expert knowledge was used, and the corpus was not annotated.

The SUMTIME team estimate<sup>6</sup> that very approximately 12 person months went directly into developing the SUMTIME microplanner and realiser, and 24 on generic activities including expert consultation, some of which also benefited the microplanner/realiser. The *p*CRU wind forecaster was 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 would be 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.

<sup>6</sup>Personal communication with E. Reiter and S. Sripada.

## 4 Discussion

The main contributions of the research described in this paper are: (i) a generation methodology that improves significantly 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 very high-quality text. Results also show that (i) a preference for shorter realisations is harmful in NLG; and that (ii) linguistically literate, probabilistic NLG can outperform shallow statistical methods, in terms of quality and efficiency.

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 estimates how well a particular base generator performs on its own. However, different base generators affect the performance of the different modes in different ways. 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), the improvements made little difference to the results for *p*CRU-2gram (up from BLEU-4 0.45 to 0.5), but greatly improved the performance of the greedy mode (up from 0.43 to 0.64).

The current situation in NLG recalls NLU in the late 1980s, when symbolic and statistical NLP were entirely separate research paradigms, a situation memorably caricatured by Gazdar (1996). In the early 1990s, NLU quickly moved towards a paradigm merger, realising that symbolic NLP lacked the efficiency and robustness that probabilistic NLP could provide, which in turn would benefit from the accuracy and subtlety of symbolic NLP (Gazdar, 1996, p. 98). 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), and this trend looks set to continue. The lesson to be learnt from NLU and MT appears to be

that higher quality results when the symbolic and statistical paradigms join forces.

## 5 Conclusions

The research reported in this paper is intended to be a first step in the direction of a merger between the statistical and the linguistic-symbolic paradigms in NLG. 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 very high quality can be produced by probabilistically driven NLG, provided it has linguistic grounding. We have seen furthermore that *p*CRU speeds up development and improves reusability of systems, and in some modes is more efficient and less brevity-biased than existing *n*-gram techniques.

If NLG is concerned with generating language from non-language representations of content or meaning, then it is currently a small field: large parts of document summarisation, machine translation and human-computer dialogue no longer use non-language representations of content. If NLG is to make a come-back in these areas it needs to move away from brittle and disposable systems towards more robust and reusable methods, something that linguistically literate, probabilistic methods can help achieve.

## Acknowledgments

The research reported in this paper is part of the CoGenT project, an ongoing research project supported under UK EPSRC Grant GR/S24480/01.

## 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, Information Technology Research Institute, 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, NLTG, CMIS, University of Brighton.
- 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. Goldberg, N. Driedger, and R. Kittredge. 1994. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2):45–53.
- E. Hovy. 1988. *Generating Natural Language under Pragmatic Constraints*. Lawrence Erlbaum.
- P. Isabelle. 1984. Machine translation at the TAUM group. In M. King, editor, *Machine Translation Today: The State of the Art*. Edinburgh University Press.
- 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.
- S. Riezler and J. T. Maxwell III. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proc. ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, pages 57–64.

- S. Sripada, E. Reiter, J. Hunter, and J. Yu. 2002. SUMTIME-METEO: Parallel corpus of naturally occurring forecast texts and weather data. Technical Report AUCS/TR0201, Computing Science Department, University of Aberdeen.
- 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 natural language generation. In *Proceedings of NAACL'01*, pages 1–8.
- M. White. 2004. Reining in CCG chart realization. In A. Belz, R. Evans, and P. Piwek, editors, *Proceedings INLG'04*, volume 3123 of *LNAI*, pages 182–191. Springer.