Natural Language Engineering **1** (1): 1–26. Printed in the United Kingdom

Automatic Generation of Weather Forecast Texts Using Comprehensive Probabilistic Generation-Space Models

ANJA BELZ

Natural Language Technology Group
School of Computing, Mathematical and Information Sciences
University of Brighton, Lewes Road, Brighton BN2 4GJ, UK.

(*Received 10 August 2006; revised 31 May 2007*)

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 were 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 scored more highly by human judges than forecasts written by experts.

1 Background

Language is generated in many NLP applications, including MT, document summarisation and dialogue systems. Researchers in the area of natural language generation (NLG) tend to define their task in such a way that it involves an abstract representation of meaning, something that large parts of MT, summarisation and dialogue do not currently involve. An influential definition of the core tasks in applied NLG was provided by Reiter and Dale (2000): content determination, discourse planning, sentence aggregation, lexicalisation, referring expression generation and linguistic realisation. While there is considerable agreement about core tasks, there is little consensus about how these tasks translate into system modules.

Building NLG systems involves considerable time and expense. Traditionally, NLG systems are carefully handcrafted as deterministic decision-makers that make decisions locally, at each step in the generation process. Decisions are encoded as generation rules with conditions for rule application (often in the form of if-then rules or rules with parameters to be matched), usually on the basis of corpus anal-

ysis and expert consultation. Great emphasis has been placed on the importance of carrying out corpus analysis manually. Reiter and Dale's influential paper (1997) recommended that NLG systems be built largely "by careful analysis of the target text corpus, and by talking to domain experts" (p. 74, but also pp. 58, 61, 72 and 73), acknowledging that this may involve time and effort:

> You are going to have to spend a considerable amount of time learning about the domain, poring over and analysing the corpus, and discussing your observations with the domain experts [...]. The amount of effort required should not be underestimated; [...]. (Reiter and Dale, 1997, p. 73)

To make this task manageable, the NLG system builder often has to eliminate much of the variation in the corpus, since writing generation rules with conditions for application precise enough to enable deterministic language generation becomes harder the more variation has to be accounted for. McKeown *et al.* (1994) comment on "the tremendous variety in the possible sentences for each message type with respect to sentence structure and lexical choice" (p. 9) that the PLANDoc project team found in their domain corpus, and describe how one of "the two overriding practical considerations" was found to be "the need for a bounded sublanguage" (p. 8) which eliminated most of the variation found in the corpus.

Re-use of systems or even system components is almost non-existent in NLG. This is in large parts due to a lack of agreement about modules and interfaces, but a major contributing factor is also the prevalence of rule-based symbolic approaches. The very nature of such generators makes their re-use difficult, even where module and interface specifications are agreed: required content, discourse structure, degrees of formality, technicality, etc., will vary from one domain and application to the next, and if they are encoded in handcrafted generation rules with fine-grained conditions for rule application, they will have to be rewritten by hand for every new application.

Re-usability is limited even in the case of surface realisers that are applied after all content choices, and most lexical and deep syntactic choices have been made. Wide-coverage surface realisers such as PENMAN/NIGEL (Mann and Mathiesen, 1983; Mann and Mathiesen, 1985), FUF/SURGE (Elhadad and Robin, 1996) and REALPRO (Lavoie and Rambow, 1997) that were intended to be more or less off-the-shelf plug-and-play modules tend to require a significant amount of work to fine-tune and integrate. They also tend to make substantial demands on the module supplying the inputs. For example, a case study looking at system builders' experience in using PENMAN in applications found that constructing input specifications for PENMAN was one of two tasks requiring "substantial effort" (Kasper, 1989, p. 155):

> Penman's grammar can control several hundred different (semantic) features. If the application program were required to specify values for all of these features for every sentence to be generated, the system would be too complex for most practical purposes. (Kasper, 1989, p. 153.)

To overcome these problems, a flexible interface to the surface realiser was developed which enabled application programs to supply inputs of varying granularity to the surface realiser.

Two important recent trends in NLG have developed at least in part to address the issues of development time and reusability in building NLG systems: one is an increasing use of statistical techniques, the other a range of methodologies that take a comprehensive view of the generation task.

NLG as a field was largely unaffected by the statistical revolution in NLP that began in the 1980s, but over the last decade, NLG researchers have become increasingly interested in statistical techniques. Since Langkilde and Knight's influential work on statistical surface realisation (Knight and Langkilde, 1998) which resulted in the NITROGEN and HALOGEN systems, 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 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.

A parallel development has been a rethinking of the traditional modular, pipelined 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 the probabilistic and the recent comprehensive 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, while the latter overgenerates vastly (including ungrammatical realisations), has a high computational cost (see Section 4), and the statistical model is not linguistically informed.

Existing comprehensive approaches can also be hard to use in practice, requiring manual experimentation to set constraints (ICONOCLAST), or annotated corpora to determine the cost associated with constraints (Langkilde, Marciniak and

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

Strube). Handling violability of soft constraints is problematic, and converting corpus-derived probabilities into costs associated with constraints turns straightforward statistics into an *ad hoc* search heuristic. The older approaches are not globally optimisable (PAULINE) or involve exhaustive search (ICONOCLAST).

2 Introduction and overview

The language generation process can be seen as the incremental specification of the word strings that form the outputs. In the course of the generation process, representations become increasingly specific in terms of determining the output string: inputs are less specific than intermediate representations which in turn are less specific than the outputs. All representations, input, intermediate and output, can be said to be in specificity relations with each other. Together they form the *generation space* underlying an NLG system.

The way that generation spaces are navigated can vary greatly from one NLG system to the next. Different systems allow different degrees of nondeterminism, and different mechanisms are applied to make decisions in the face of nondeterminism. In a traditional rule-based NLG system, decision-making and generation space are interleaved: rule application conditions ensure that a generation process is simply a path through the generation space from an input to an output, any nondeterminism in the rules being resolved by defaults or arbitrary selection. In a generate-and-select system such as HALOGEN, decision-making and generation space are entirely separate: a generation process is a tree leading to multiple outputs, and nondeterminism is not resolved until after generation is complete, when a decision is made in favour of one of the leaf nodes (realisations).

Building NLG systems involves encoding generation spaces and decision-making abilities which translates, at a high level of abstraction, into the tasks of (i) determining the range of variation in content, semantic, syntactic and surface forms (perhaps as found in a target corpus); and (ii) determining the conditions under which each variant is preferred over alternatives. The more time-consuming of the two tasks is likely to be *ii*: determining the range of variation in a corpus is more straightforward than determining the conditions under which a particular variant should be preferred over alternatives (see also discussion in Section 5). In traditional rule-based generators, both *i* and *ii* are created manually, whereas in an *n*-gram-model-based generate-and-select system like HALOGEN, the decision-making abilities are acquired automatically.

*P*robabilistic Context-free Representationally Underspecified (*p*CRU) language generation shares one fundamental aim with HALOGEN-type generation: to formally separate *i* and *ii*, and to completely automate *ii*, using relative frequency as the basis for decision-making. However, *p*CRU aims to provide an efficient, linguistically informed, statistically principled alternative to existing statistical and comprehensive approaches. It combines a probabilistic generation methodology with a comprehensive model of the generation space. Both generate-and-select systems and *p*CRU are faster to develop than traditional NLG systems, because their decision-making abilities are acquired entirely automatically. The core differences between

n-gram-based generate-and-select and pCRU are that in the latter, probabilistic choice informs generation as it goes along, instead of generating all alternatives and then selecting probabilistically (as in HALOGEN), and probabilities attach to the decisions that the generation space is composed of, as opposed to arbitrary word sequences (as in n-grams in general). The overall aim behind pCRU was to create an approach to generation that would have the advantages of HALOGEN-type generation as compared to traditional approaches (i.e. reduced development time and enhanced reusability), but would improve over generate-and-select generation by increasing computational efficiency and language quality.

This paper summarises the pCRU framework, which is described in more detail in two technical reports (Belz, 2004; Belz, 2006), and reports the results of experiments designed to rigorously test pCRU in practice and to determine whether it does achieve improvements in development time and reusability while sacrificing neither quality of outputs nor efficiency of generation.

The paper is structured as follows. The pCRU language generation framework is summarised in the following section, the implementation and evaluation of the pCRU weather forecast generator is described in detail in Section 4, while Section 5 discusses some of the evaluation results in more detail, also outlining directions for further research. Concluding remarks can be found in Section 6.

3 *p*CRU language generation

Probabilistic Context-free Representationally Underspecified language generation, or pCRU, is a language generation framework that was developed (in the UK 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 representations (performing different tasks in different modules). The basic idea in pCRU 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.

pCRU uses straightforward context-free technology in combination with underspecification techniques (context-free representational underspecification, or CRU; see Belz, 2004), and the separation between generation space encoding and decision-making described in the previous section is fundamental to it. The generation space is encoded as a **base generator** in the form of (i) a set G of expansion rules (of the form above) 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. 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 pCRU ($p$robabilistic 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 for the sentence 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.

Creation of base generators is described in more detail in the following section, and training of the decision-maker in Section 3.2. A software package implementing *p*CRU which includes two additional baseline generation techniques is briefly described in Section 3.3.

3.1 Defining generation spaces: pCRU base generators

Generation spaces are encoded using CRU (context-free representational underspecification; Belz, 2004) as a set of expansion rules composed of n-ary relations. Such a set of expansion rules explicitly 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 seen explicitly as being the task of incrementally specifying one or more word strings.

Generation rules are required to be context-free. A *p*CRU generation space is therefore a 4-tuple $G = (W, N, S, R)$, where W is a set of terminals, N is a set of nonterminals, $S \in N$ is the start symbol, and R is a set of production rules, where each rule is of the form $n \rightarrow \alpha$, $n \in N$, $\alpha \in (W \cup N)^*$, and W and N are disjoint.

(Non)terminals are terms $f(b_1, ..., b_n)$ where $f \in F$ is an n-ary relation with $n \geq 0$, $b_1, ..., b_n \in V \cup C$ are variables and constants, V is an alphabet of variable names, C a set of constants, and F is an alphabet of relation names.

The input to generation can be any sentential form licensed by G. An expansion rule can be applied only if a unifying substitution exists for the nonterminal to be expanded and the right-hand side of the rule. In non-probabilistic mode, the output is the set of terminal strings that the input can be expanded to under G.

Within the limits of context-freeness and atomicity of feature values, *p*CRU is neutral with respect to actual linguistic knowledge representation formalisms used to encode generation spaces. The semantics and lexicon that have been used in experiments with *p*CRU so far are based on Multiple Recursion Semantics (MRS) as used in the Lingo parser-realiser (Copestake *et al.*, 2005). Figure 1 shows a small fragment from the generation space definition for a patient information leaflets generator (based on the corpus described in Bouayad *et al.*, 2000). The idea was to create a flexible way of linking up to the Lingo realiser (which requires highly specific inputs), by abstracting over MRS representations. The *p*CRU rules generate MRS representations which are then passed on to the Lingo realiser. The rules in Figure 1 generate MRS representations of verb phrases such as *take the medicine, should take the medicine*, etc. The conventions adopted for representing Lingo-MRS semantic

TAKE_MED(H_1, E, *E.TENSE*, *E.MOOD*, *E.ASPECT.PERF*, *E.ASPECT*, X_1, *PNG.PN*$_1$, *DIVISIBLE*, X_2, H_0, H_3) →
  MEDICINE(H_1, X_2, *neut*, *PNG.PN*$_2$, H_2)
  TAKE(H_2, E, *E.TENSE*, *E.MOOD*, *E.ASPECT.PERF*, *E.ASPECT*, X_1, *PNG.PN*$_1$, *DIVISIBLE*, X_2, *neut*, *PNG.PN*$_2$, U, H_0, H_3)
MEDICINE(H_1, X, *neut*, *PNG.PN*, H_3) →
  nq_the_q_rel(H_1, X, *PNG.PN*, *DIVISIBLE*, H_2, H_3)
  qq_medicine_n_relqq(H_2, X, *neut*, *PNG.PN*)
TAKE(H_1, E, *present*, *E.MOOD*, -, -, X_1, *PNG.PN*$_1$, *DIVISIBLE*, X_2, *PNG.GEN*, *PNG.PN*$_2$, U, H_0, H_2) →
  imp_m_rel(H_0, H_2)
  qq_take_v_1_relqq(H_1, E, *present*, *E.MOOD*, -, -, X_1, *PNG.PN*$_1$, *DIVISIBLE*, X_2, *PNG.GEN*, *PNG.PN*$_2$, U)
TAKE(H_1, E, *present*, *indicative*, -, -, X_1, *PNG.PN*$_1$, *DIVISIBLE*, X_2, *PNG.GEN*, *PNG.PN*$_2$, U, H_0, H_2) →
  prpstn_m_rel(H_0, H_2)
  qq_take_v_1_relqq(H_1, E, *present*, *indicative*, -, -, X_1, *PNG.PN*$_1$, *DIVISIBLE*, X_2, *PNG.GEN*, *PNG.PN*$_2$, U)
TAKE(H_1, E_1, *present*, *ind_or_mod_subj*, -, -, X_1, *PNG.PN*$_1$, *DIVISIBLE*, X_2, *PNG.GEN*, *PNG.PN*$_2$, U, H_0, H_2) →
  prpstn_m_rel(H_0, H_2)
  nq_should_v_rel(H_1, E_1, *present*, *ind_or_mod_subj*, -, -, H_3)
  qq_take_v_1_relqq(H_3, E_2, *no_tense*, *indicative*, -, -, X_1, *PNG.PN*$_1$, *DIVISIBLE*, X_2, *PNG.GEN*, *PNG.PN*$_2$, U)

Fig. 1. Fragment of *p*CRU generation space for patient information leaflets. The rules generate a set of MRS expressions which when fed to the Lingo realiser generate 42 different variants of pill-taking instructions (controlled by the arguments), including *take the medicine, you take the medicine, the patient takes the medicine, patients take the medicines, you should take the medicine*, etc.

relations in CFG rules are described in detail in a previous report (Belz, 2004). Briefly, H-variables represent labels and pointers, X-variables are entity variables, E-variables are event variables. Lower case arguments are constant strings. Upper-case relations are non-terminal relations, lower-case relations are terminals. Double-quoted terminals (starting and ending with qq) are lexical items, those ending in *m_rel* are message type relations (e.g. imperative, proposition, etc.).

When building generation space definitions, the idea is that existing resources, in particular for surface realisation, are reused, to the extent that they can be converted into rules of the form above. For the patient information leaflet domain, the surface rules were adapted from the Lingo grammar. For the rather simpler weather forecasting domain (Section 4), rules were written from scratch.

3.2 Selection among alternatives: pCRU decision-making

The *p*CRU (*p*robabilistic 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 c for each individual generation rule $N \rightarrow \alpha$ from the multi-treebank, by adding $1/n$ to c for every occurrence of the rule in a set of derivation trees for a sentence, where n is the total number of alternative derivation trees for the sentence; convert counts into probability distributions over alternative rules, using add-1 smoothing[2] and standard maximum likelihood estimation:

$$p(N \rightarrow \alpha) = \frac{c(N \rightarrow \alpha)}{\sum_{i:N \rightarrow \alpha_i \in R} c(N \rightarrow \alpha_i)} \quad (1)$$

During generation, inputs are expanded by applying expansion rules as described in Section 3.1, and the pCRU probability distribution is used in one of three ways to inform the generation process:

1. **Greedy generation:** apply only the most likely rule to expand each nonterminal; this means making the single most likely decision at each choice point in a generation process which is not guaranteed to result in the most likely generation *process*, but the computational cost is very low.
2. **Viterbi generation:** apply all expansion rules to each nonterminal to create the generation forest for the input, then do a Viterbi search of the generation forest; this maximises the joint likelihood of all decisions taken in the generation process. This does select the most likely generation process[3], but is considerably more expensive.
3. **Greedy roulette-wheel generation:** select a rule to expand a nonterminal according to a non-uniform random distribution proportional to the likelihoods of expansion rules. E.g. if there are two alternative rules $D1$ 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%.

3.3 The pCRU-1.0 generation package

The technology described in the two preceding sections has been implemented in a software package released as pCRU-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

[2] This is equivalent to Bayesian estimation with a uniform prior probability on all decisions, and is entirely sufficient for present purposes given the small vocabulary and the good coverage of the data.

[3] NB, *not* the most likely string.

This base generator is then automatically 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. Two additional modes are provided:

1. **Random**: ignoring pCRU probabilities, randomly select generation rules.
2. **N-gram**: ignoring pCRU probabilities, generate set of alternatives and select the most likely according to the n-gram language model (as in HALOGEN).

The random mode serves as an absolute baseline for generation quality: a trained generator must be able to do better, otherwise the probabilities are not doing any work, and all the work is done by the base generator. 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 pCRU probabilities should be able to produce realisations faster (because it uses probabilities integrated into the generation space model, rather than looking them up in a separate model).

4 Building and evaluating a *p*CRU wind forecast generator

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 NLP researchers, and a number of weather forecast systems have been created, of which the two best known are perhaps METEO (Isabelle, 1984) and FOG (Goldberg *et al.*, 1994). METEO translates weather forecasts from English to French and vice versa. It produced the first machine-translated weather forecast on May 24th, 1977, and has evolved over the years to cope with more than 90% of the workload of the translation team that uses it[4]. FOG is a bilingual system that generates English and French marine forecasts from a common content representation, and was one of the first commercially used NLG systems.

A recent example of weather forecast generation research is the SUMTIME project (Reiter *et al.*, 2005) which developed an NLG system that generates marine weather forecasts for offshore oil rigs from numerical forecast data produced by weather simulation programs. The SUMTIME system has two modules: a content-determination module and a microplanning and realisation module (the system can be run without the content-determination module, taking content representations as inputs, and is then called SUMTIME-Hybrid). The corpus developed for SUMTIME is used in the experiments below, and results are compared to SUMTIME-Hybrid outputs.

While the restrictiveness of the domain language in meteorology is an advantage, it is quantitative rather than qualitative as far as lexicon and syntax are concerned, and scaling up to less restrictive sublanguages would involve the addition of more lexical items and more syntactic structures, but no change to generation methodology. However, weather forecasts tend to be short sequences of statements with few

[4] According to MT News International, the Newsletter of the International Association for Machine Translation, Issue no. 17, June/July 1997, ISSN 0965-5476.

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

05/06     SSW     18      22      27      3.0     4.8     SSW     2.5     9
05/09     S       16      20      25      2.7     4.3     SSW     2.3     9
05/12     S       14      17      21      2.5     4.0     SSW     2.2     9
05/15     S       14      17      21      2.3     3.7     SSW     2.2     8
05/18     SSE     12      15      18      2.4     3.8     SSW     2.3     8
05/21     SSE     10      12      15      2.4     3.8     SSW     2.4     8
06/00     VAR     6       7       8       2.4     3.8     SSW     2.4     8
...
```

Fig. 2. Meteorological data file for 05-10-2000, a.m. (the names of oil fields have been anonymised).

```
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
  50M:          SSW 20-26 GRADUALLY BACKING SSE THEN FALLING
                VARIABLE 08-12 BY LATE EVENING
...
```

Fig. 3. Wind forecast for 05-10-2000, a.m. (the names of oil fields have been anonymised).

discourse connectives or anaphoric references, which does make them much simpler to generate at the discourse level than most text types.

4.1 Data

The SumTime-Meteo corpus was created by the SumTime project team in collaboration with wni Oceanroutes (Sripada *et al.*, 2002). The corpus was collected by wni Oceanroutes from the commercial output of five different (human) forecasters, and each instance in the 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 (and sometimes additional resources). The experiments below focused on the part of the forecasts that predicts wind characteristics for the next 15 hours.

Only the data file type that contains (virtually all) the information about wind parameters (the .tab file type) was used. Figure 2 shows an example .tab file and Figure 3 shows the corresponding wind forecast. In Figure 2, the first column is the day/hour time stamp, the second the wind direction predicted for the corresponding time period; the third the wind speed at 10m above the ground; the fourth the gust speed at 10m; and the fifth the gust speed at 50m. The remaining columns contain wave data.

The mapping from time series data to forecast is not straightforward (even when all three data files are taken into account). For example, in Figures 2 and 3, how

the wind speeds in the table map to speed ranges in the forecast seems to be largely down to forecasters' individual preferences.

For the experiments below, a version of the SumTime-Meteo corpus was created that contains pairs of wind statements and the wind data that is actually included in the statement, so that generators could be run with the same inputs as SumTime-Hybrid. The wind data is a vector of time stamps and wind parameters, and was 'reverse-engineered', by automatically aligning wind speeds and wind directions in the forecasts with time-stamps in the data file. In order to do this, wind speed and directions in the data file have to be matched with those in the forecast. This was not straightforward, because often there is no exact match in the data file for the wind speeds and directions in the forecast. The strategy adopted was the same as in the SumTime work, in order to make the systems comparable[5]. The following is an example input/output pair from the final corpus, consisting of the meteorological data and corresponding wind forecast:

Data:	`1 SSW 16 20 - - 0600 2 SSE - - - - NOTIME 3 VAR 04 08 - - 2400`
Forecast:	`SSW 16-20 GRADUALLY BACKING SSE THEN FALLING VARIABLE 4-8 BY LATE EVENING`

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, see Table 1).

4.2 The base generator

The *p*CRU base generator for the SumTime domain was written semi-automatically as a set of generation rules with atomic arguments that convert an input vector of numbers and symbols in steps to a set of NL forecasts. The automatic part was analysing the entire corpus with a set of simple chunking rules that split wind statements into wind direction, wind speed, gust speed, gust statements, time expressions, transition phrases (such as *increasing to*), pre-modifiers (such as *less than* for numbers, and *mainly* for wind direction), and post-modifiers (e.g. *in showers*). Chunking the corpus in this way resulted in several sets of phrases: the set of all wind directions, the set of all wind speeds, the set of all time expressions etc. These were converted into preterminal and lexical rules expanding e.g. a nonterminal representing 'wind direction change verb' to *backing*, and a nonterminal representing 'speed transition change verb' to *easing*.

The manual work on the base generator consisted in writing the chunking rules themselves, and higher-level rules that combine different sequences of preterminals into larger components, taking care of text structuring, aggregation and elision. The

[5] Reiter *et al.* selected the time stamp of the data in the table that most closely matched the data in the forecast, and if there was not a close enough match, they derived a time stamp from the time expression in the forecast, and finally, if that could not be done with enough confidence, then time was left unspecified.

higher-level rules were based on an interpretation of wind statements as sequences of fairly independent units of information ('segments'), each containing as a minimum a wind direction or wind speed range, and as a maximum all the chunk types listed above. The context encoded in the rules was the position of a unit of information in a wind statement; whether a wind statement contained wind direction (only), wind speed (only), or both; whether the change in wind direction was clockwise or anti-clockwise; whether change in wind speed was an increase or a decrease. The final base generator takes as inputs number vectors of length 7 to 60, and has a large amount of non-determinism. For the simplest input, it generates 5 alternative realisations. For the most complex input, it would generate 1.6×10^{31} alternatives.

As an illustration of the role of the higher-level rules, consider the following example (note that this is not a rule in the generation grammar, but a summary of multiple derivation processes, involving multiple rule applications):

$$Segment(2, -1, up, counterclock, slow, ssw, 22, 28, n, n, n) \stackrel{*}{\Longrightarrow}$$

gradually backing SSW and gradually increasing 22-28
gradually backing and increasing SSW 22-28
backing/increasing gradually SSW 22-28
backing SSW increasing 22-28
backing gradually SSW 22-28

The arguments to the *Segment* relation are part of an input vector of weather data, augmented by contextual information. The weather data arguments represent wind direction (ssw), minimum wind speed (22), maximum wind speed (28), minimum/maximum gust speed and time stamp (which in this example are all unspecified, represented by n). The other arguments encode the contextual information that this is the second segment (2), but not the last (-1), that wind speed has increased compared to the preceding segment (up), that the wind direction has changed counter-clockwise ($counterclock$), and that the changes are gradual ($slow$). This input segment generates, among many others, the five word strings shown above. While the generation rules allow the first word string, it is not one that meteorologists would actually use, preferring to aggregate into one of the shorter forms above (all of which do occur in the corpus). Quite frequently, one of the verbs describing type of change is dropped (*increasing* in the the last word string above), or the manner adverb is dropped (*gradually* in the second last string above).

Figure 4 shows a fragment of the base generator which generates phrases describing gusts. The grammar rules have been somewhat simplified here, for better readability. The topmost *Gusts* relation takes as arguments information about maximum gust speed, minimum gust speed (if any), and presence of showers/thunderstorms (if any). Arguments with initial upper case letters are variables, arguments with initial lower case are constants. The type definitions (not shown here) define Nv to be any positive integer, N to be Nv or unspecified (n), $ST \in \{s, t, n\}$ which encodes presence of showers, thunderstorm, or the unspecified value. Note that the rules on their own merely list the variants found in the corpus, they do not enable decisions to be made among the variants. Full details of CRU and representational conventions can be found in (Belz, 2004).

$Gusts(Nv_1, N_2, n) \rightarrow GustCore(Nv_1, N_2)$
$Gusts(Nv_1, N_2, ST) \rightarrow GustCore(Nv_1, N_2)\ GustPostMod(ST)$

$GustCore(Nv, n) \rightarrow GustTrans\ Num(Nv)$
$GustCore(Nv_1, Nv_2) \rightarrow GustTrans\ Num(Nv_1) - Num(Nv_2)$

$GustTrans \rightarrow$ gusting
$GustTrans \rightarrow$ gusts
$GustTrans \rightarrow$ gusts to
$GustTrans \rightarrow$ in gusts
$GustTrans \rightarrow$ risk gusts to
$GustTrans \rightarrow$ with gusts
$GustTrans \rightarrow$ with gusts to

$GustPostMod(s) \rightarrow$ in any showers
$GustPostMod(s) \rightarrow$ in or near showers
$GustPostMod(s) \rightarrow$ in showers
$GustPostMod(t) \rightarrow$ in any thunderstorm
$GustPostMod(t) \rightarrow$ in any thunderstorms
$GustPostMod(t) \rightarrow$ in any thundery showers

Fig. 4. Fragment of generation space for SumTime weather forecast generators which generates gust phrases (rules have been simplified for readability).

4.3 Training

The corpus was divided at random into training and testing data at a ratio of 9:1. The training set was multi-treebanked (see Section 3.2 above) with the base generator, and the multi-treebank was then used to create the probability distribution for the base generator (as described in Section 3.2). A back-off 2-gram model with Good-Turing discounting and no lexical classes was also created (using the SRILM toolkit; Stolcke, 2002) from the training set. *p*CRU-1.0 was 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 1) indicated that five repeats were sufficient.

The following is a subset of the rules from Figure 4 together with their log probabilities as obtained by training in one of the five runs. The shortest phrase at the top is by far the most likely (as e.g. in NNE 22-26 GUSTS 36 BACKING NNW 14-18 BY EVENING):

$-0.1513\ GustTrans \rightarrow$ gusts
$-2.3978\ GustTrans \rightarrow$ with gusts to
$-4.1026\ GustTrans \rightarrow$ risk gusts to
$-4.1026\ GustTrans \rightarrow$ with gusts
$-4.7957\ GustTrans \rightarrow$ gusts to
$-5.4889\ GustTrans \rightarrow$ gusting
$-5.4889\ GustTrans \rightarrow$ in gusts

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`
Human1	`SSW'LY 16-20 GRADUALLY BACKING SSE'LY THEN DECREASING VARIABLE 4-8 BY LATE EVENING`
Human2	`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`
*p*CRU:	
-greedy	`SSW 16-20 BACKING SSE FOR A TIME THEN FALLING VARIABLE 4-8 BY LATE EVENING`
-roulette	`SSW 16-20 GRADUALLY BACKING SSE AND VARIABLE 4-8`
-viterbi	`SSW 16-20 BACKING SSE VARIABLE 4-8 LATER`
-2gram	`SSW 16-20 BACKING SSE VARIABLE 4-8 LATER`
-random	`SSW 16-20 AT FIRST FROM MIDDAY BECOMING SSE DURING THE AFTERNOON THEN VARIABLE 4-8`

Fig. 5. Example input with corresponding outputs by all systems and some meteorologists (for 5 Oct 2000).

Rule probabilities are applied in different ways by the different *p*CRU decision-makers, whose outputs can be very different. Table 5 shows an example data input and *p*CRU generator outputs, along with the corresponding forecasts from the corpus, from two additional expert human forecasters, from SumTime-Hybrid, and the *p*CRU baseline modes.

4.4 Evaluation

4.4.1 Evaluation methods

The two automatic metrics used in the evaluations, NIST[6] and BLEU[7], have been shown to correlate well with expert judgments (Pearson correlation coefficients 0.82 and 0.79 respectively) in the SumTime domain (Belz and Reiter, 2006). BLEU (Papineni *et al.*, 2002) was developed for MT, and is a precision metric (with brevity penalty) that assesses the quality of a translation in terms of the proportion of its word n-grams ($n \leq 4$ 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. NIST scores are ≥ 0, but the upper limit is reference set dependent. Some research has shown NIST to correlate with human judgments more highly than BLEU (Doddington, 2002; Riezler and Maxwell III, 2005; 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

[6] `http://cio.nist.gov/esd/emaildir/lists/mt_list/bin00000.bin`
[7] `ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl`

Table 1. NIST-*5 and* BLEU-*4 scores for training and test sets (average variation from the mean).*

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

subjects[8]. 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 all five corpus authors. In the two human experiments, NIST and BLEU scores were computed against sets of multiple reference texts (two for each date in Experiment 1, and three for each date 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.

4.4.2 Comparing different generation modes

Table 1 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 1. For the training set, significance test 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-roulette, and *p*CRU-viterbi vs. *p*CRU-roulette.

[8] Belz and Reiter, in preparation.

Table 2. *Average scores for handcrafted system and two best* pCRU*-systems from two human experiments.*

	Experiment 1	Experiment 2
SumTime-Hybrid	3.82 (1)	4.61 (2)
*p*CRU-greedy	3.59 (2)	4.79 (1)
*p*CRU-roulette	3.22 (3)	4.54 (3)

NIST scores depend on test set size, and are necessarily lower for the (smaller) test set, but the BLEU-4 scores indicate that performance was slightly worse on test sets (as would be expected). 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. Subjects in the human-based evaluations agreed with the NIST-5 rankings exactly.

These comparative results for the different *p*CRU-modes provide a different picture from those previously reported for an earlier version of the *p*CRU wind forecast generator (Belz, 2005), where *p*CRU-2gram consistently outperformed all other modes, and there was little difference between the Viterbi and greedy modes. The difference between the earlier results and those reported here are due to the improvements in the definition of the generation space which made *p*CRU probabilities more fine-grained and discriminating. As predicted in the earlier report (Belz, 2005, p. 21), the improvements made little difference to the results for *p*CRU-2gram, but improved the performance of the greedy modes substantially (see also discussion in Section 5). An unexpected effect was that the Viterbi mode did not improve more and still lags very slightly behind *p*CRU-2gram. A possible explanation may be that the length bias of the Viterbi mode (see Section 4.4.6 below) cancels out any improvements in the generation rules.

4.4.3 Text quality against handcrafted system

The *p*CRU modes were also evaluated against the SumTime-Hybrid system. Table 2 shows averaged evaluation scores by subjects in the two independent experiments described above. Altogether 6 and 7 systems were evaluated in Experiments 1 and 2, respectively. The differences between the *p*CRU and SumTime-Hybrid scores shown here were not significant when subjected to the Tukey Test with multiple-test correction, 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 systems. The expert forecast readers

Table 3. *Average scores for human-written forecasts and best* pCRU*-system from two human experiments.*

	Experiment 1	Experiment 2
*p*CRU-greedy	3.59	4.79
Corpus	3.22	4.50
Meteorologist	3.03	–

scored SumTime-Hybrid more highly in the first experiment, and *p*CRU-greedy more highly in the second, by similar margins.

4.4.4 Text quality against human forecasters

Table 3 shows average scores given by the expert evaluators in the two experiments to *p*CRU-greedy, the corpus texts, and texts written by another (human) forecaster. While the expert evaluators scored *p*CRU-greedy higher than all human-written texts in both experiments, the differences are not significant after applying the multiple-test correction. However, while in each experiment separately, statistical significance could not be shown, in combination the scores provide evidence that the evaluators considered *p*CRU-greedy texts better than the human-written texts.

4.4.5 Computing time

The *p*CRU base grammar encodes a large generation space. For maximum length inputs there are up to 1.6×10^{31} different realisations. For efficiency reasons, the base grammar was actually implemented in such a way as to ensure only linear increase in generation complexity with increasing length: the effect of this is that each input segment is expanded separately. As a result, the total computing times and the differences between the four non-random generation modes are not as large as reported for previous experiments[9]. The following table shows seconds taken to generate one forecast[10], 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)

[9] Where the 2gram mode took about 7 times as long as the Viterbi mode, which took 9–13 times longer than the greedy mode (Belz, 2005).
[10] On a Viglen XX225 Server with 2 3.06Ghz Intel Xeon CPUs, and 4Gb Ram.

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[11].

4.4.6 Brevity bias

There are substantial differences in the average length of the forecasts generated under the control of the different decision-makers, and it is those forecasts with an average length substantially longer (*p*CRU-random) or shorter (*p*CRU-2gram and *p*CRU-viterbi) than the corpus that have the worst evaluation results. The average number of words in the forecasts generated by the different systems was as follows:

*p*CRU-random:	19.43
SumTime:	12.39
*p*CRU-greedy:	11.51
Corpus:	*11.28*
*p*CRU-greedy-roulette:	10.48
*p*CRU-2gram:	7.66
*p*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 has an almost absolute preference of shorter over longer strings, and so produces the second shortest strings. The Viterbi generator does not prefer shorter strings, but does prefer shorter derivations, and there is a correlation between string length and derivation length. The greedy generators do not have a built-in preference for shorter strings or derivations, and they achieve the closest matches to the average forecast length in the corpus.

The reason why n-gram models have a built-in bias towards shorter strings is that 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.

The n-gram model's bias towards shorter strings is an example of a general case: whenever the likelihood of a unit that can vary in length (e.g. sentence) is modeled in terms of the joint probability of length-invariant smaller units, those units that are composed of fewer smaller units are more likely. Another example is PCFG parsing where the likelihood of a parse is the joint probability of the expansion rules it is composed of, and therefore parse trees with fewer nodes are more likely.

[11] The Viterbi and the 2-gram generator are implemented identically, except for the way in which they obtain the probabilities to be inserted into the generation forest: the former uses the (internal) *p*CRU rule probabilities, and the latter looks up an external n-gram model.

The possibility of counteracting this bias has been investigated. In parsing, Magerman and Marcus (1991) and Briscoe and Carroll (1993) used the geometric mean of probabilities instead of the product of probabilities. In later work, Briscoe and Carroll (1992) use a normalisation approach, the equivalent of which for n-gram selection would be to 'pad' shorter alternatives with extra 'dummy' words up to the length of the longest alternative, and to use the geometric mean of the probabilities assigned by the n-gram model to the non-dummy words as the probability of any dummy word. It has been observed that such methods turn a principled probabilistic model into an *ad hoc* scoring function (Manning and Schuetze, 1999, p. 443). It certainly means getting rid of the n-gram model's particular independence assumptions, without replacing them with a statistically motivated alternative.

To some extent, a preference for shorter strings is hardwired into PCFGs which assign a disproportionately large part of the probability mass to shorter strings. It appears that the greedy selection strategies overcome this preference.

In some applications, the n-gram model's preference for shorter strings is irrelevant: e.g. in speech recognition (where n-gram models are used widely) the alternatives among which the model must choose are always of the same length. In language generation, alternative ways of expressing the same meaning can vary greatly in length, and a generation method needs to be able to produce outputs of similar length to those in the target corpus. The results presented in this paper indicate that generation methods that produce shorter outputs than the target outputs that the method was trained on (clearly the case in pCRU-2gram and pCRU-viterbi) are less successful than methods which achieve a good length match[12].

4.4.7 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 the SumTime project, these were manually encoded on the basis of careful corpus analysis and consultation with writers and readers of marine forecasts. In the pCRU 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[13] 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 pCRU wind forecaster was built in less than one person 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

[12] In some NLG domains, short texts may be desirable, but then the target corpus will also consist of short texts.

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

that it would take another two person months to equip the *p*CRU forecaster with these capabilities.

This is not to say that these two figures are exactly comparable, or that the two research efforts resulted in exactly the same thing. The main point is that the SUMTIME system did come with a substantial price tag attached. Cost is moreover not restricted to the initial development time. Every time the company using the SUMTIME system changes its house style or coverage needs to be extended, the cost in terms of expert time continues to rise.

The *p*CRU approach allows control over the trade-off between cost and quality in building the initial system, which can then be adapted to new styles in one of two ways: by retraining the decision-maker, and by extending the base generator. Retraining is done fully automatically, by simply training a new decision-maker on a new corpus, whereas the base generator is extended manually by adding new rules. Retraining is sufficient if adaptation is e.g. to a new house style in the same domain, as was shown for the SUMTIME domain in a previous report (Belz, 2005b). Extending the coverage of the base grammar is also somewhat simpler than in traditional NLG generation grammars, because rules for extending coverage can simply be added without changing the rest of the grammar. These two adaptation methods are discussed in more detail in the following section.

5 Discussion and further research

In interpreting the results presented in this paper, it is important to bear in mind that they were obtained in a domain that is in some ways ideal for automatically generating language (and for MT, see beginning of Section 4): the sets of words and of syntactic structures are small, the texts are one or two sentences short, and there is hardly any discourse-level generation to deal with. As pointed out in Section 4, while scaling up the lexicon and grammar should be straightforward, it remains to be seen whether more complex discourse structuring and modeling of discourse-level phenomena can be achieved with this approach. Having said that, many applied NLG systems are for domains with similarly restricted sublanguages and short texts, and it is for such systems, rather than wide-coverage tools, that the *p*CRU approach is intended.

One of the stronger claims in this paper is that using *p*CRU cuts down on development time, compared to building an equivalent system entirely by hand. As discussed in the background and introduction sections above, the tasks involved in building applied NLG systems can be grouped into (i) discovering and encoding the variation found in the target corpus (such as the range of expressions for the same message type), and (ii) discovering and encoding the reasons and conditions for selecting one of the variants (such as why/when to select a particular expression for a given message type). Of the two, *ii* is by far the more time-consuming. In traditional manual NLG system building, *i* and *ii* tend to be interleaved and encoded together as generation rules with fine-grained conditions for rule application, often in the form of parameterised or if-then rules. In *p*CRU, *i* is encoded as a non-deterministic generation space, and *ii* as the automatically trainable (hence

adaptable) decision-maker. The saving in development time is therefore entirely in not having to encode *ii* manually. While in the weather forecast text generation domain, it was possible to build *i* semi-automatically, in more complex domains, encoding *i* will be a more involved, time-intensive process.

A related claim is that *p*CRU systems are more easily adaptable. *p*CRU systems can be adapted in two ways: one is extending the coverage of the generation space, and the other is retraining the decision-maker. If the new style or domain that the generator is to be adapted to is already covered by the generation space grammar, then adaptation can be done fully automatically. This is an issue that was explored in some detail in a previous report (Belz, 2005b) which showed that the five meteorologists who contributed to the SUMTIME corpus had clearly distinguishable styles (or idiolects), and that *p*CRU enabled the system builder to switch freely between the styles, simply by retraining on the appropriate subsection of the corpus, without incurring any manual overhead.

Extending the coverage of the generation space grammar is also somewhat simpler than in traditional NLG generation grammars, because rules for extending coverage can simply be added without changing the rest of the grammar, whereas adding rules with application conditions requires checking (and possibly adapting) other rules for possible rule interaction.

A third, somewhat more tentative, claim is that using generation models that are both rule-based and probabilistic to inform language generation produces better results than using either shallow statistical models or non-probabilistic rule-based models. 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 adaptability, 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 attaching probabilities to linguistically meaningful units and structures can outperform the shallow probabilities of *n*-gram generation techniques as well as symbolic NLG systems.

Some of the issues raised in this paper might benefit from further clarification. One is that *p*CRU is intended as a method for building entire generation systems (albeit excluding non-linguistic content determination), not just surface realisers. The *p*CRU weather forecast text generators described in this paper do more than surface realisation: while they do not perform data mining on the time-series data output by the weather simulation programs, their input is still a vector of numbers, and some typical strategic and early tactical generation tasks need to be performed, including some content determination, text planning, aggregation and elision, as well as lexical and syntactic choice. Using a single statistical model of the entire generation space (rather than several separate models, or a single model just for surface generation as in work by Marciniak and Paiva & Evans) is new in NLG.

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 pre-

vious experiments (Belz, 2005) encoded a less structured generation space and the set of concepts was less fine-grained (e.g. it did not distinguish between an increase and a decrease in wind speed), and therefore it lacked some information necessary for deriving conditional probabilities for lexical choice (e.g. *freshening* vs. *easing*). As predicted in the earlier report (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).

Somewhat surprisingly, the greedy generation modes (making locally optimal decisions) performed much better than the Viterbi mode (which makes globally optimal decisions). Together with the poor performance of the *n*-gram mode, this is an indication that a preference for shorter realisations relative to corpus text length (a property shared by the *n*-gram and Viterbi modes) is harmful in NLG. There is an interesting difference between the two greedy modes: the simple greedy mode is liked by both human evaluators and automatic evaluation metrics, whereas the greedy roulette-wheel mode is ranked nearly as low as the random mode by automatic metrics (except for NIST), but ranked highly by the human evaluators. The simple greedy mode is necessarily scored higher than the greedy roulette-wheel mode by automatic metrics that straightforwardly reward maximal string similarity (such as BLEU), because it always makes the most likely (most frequently observed) decision, and therefore maximises similarity with the observed corpus, so that when evaluated on similar material, the similarity of its outputs with the test material is very likely to be higher than that of the outputs of the greedy roulette-wheel mode which varies choice among several frequently observed decisions. This points to a serious flaw in existing word-string-similarity based metrics, as discussed in an earlier report (Belz and Reiter, 2006).

Another surprising aspect of the work presented here is that the probabilistic models were all trained on unannotated, or 'raw' corpora, and that this was sufficient to obtain high-quality results. An intuitive explanation of why this works is that in contrast to parsing, it does not really matter *how* a text is generated (derived), because the text itself is the output (whereas in parsing, how the text is derived, its parse, is the output). While results show that high-quality generators can be created by training on corpora with no annotation whatsoever, it remains to be seen just how much can be learned from unannotated corpora, and to what extent this positive result can be reproduced in other, less restricted, domains.

6 Conclusions

The starting point for the research presented in this paper was the aim to develop an approach to NLG that would reduce development time and increase reusability of systems and components, compared to traditional handcrafted NLG. The first and most fundamental methodological decision was to strictly separate generation space (unchanging) and decision-making (different from one domain to the next), and to base decision-making on an automatically estimated probabilistic model. To this extent, *p*CRU is similar to existing *n*-gram-based generate-and-select NLG such as HALOGEN. However, the second aim for the approach was to reduce computational

expense and improve the quality of generated language compared to n-gram-based generate-and-select NLG. The second fundamental methodological choice was therefore in favour of a structured probabilistic model that would make it possible (i) to make probabilistically informed decisions during generation, thus avoiding the far greater expense from deferring decisions until after generation; and (ii) to have probabilities attached to linguistically meaningful units and actual decisions, thus improving language quality.

The results presented in this paper show that the *p*CRU generation methodology achieves the above aims: (i) it improves substantially on development time and reusability compared to traditional hand-crafted systems; (ii) it produces outputs (at least in the weather domain) that were judged better than those produced by n-gram-based generate-and-select NLG techniques by all human and automatic evaluation tests that were applied; and (iii) it is computationally more efficient than generate-and-select NLG. These results do have to be interpreted in the simplifying context of weather forecast texts, and future research will have to demonstrate the feasibility of *p*CRU for more complex domains and/or wider coverage.

There is evidence from neighbouring research fields that purely symbolic and purely statistical approaches are ultimately superseded by approaches that combine aspects of both. In the late 1980s, symbolic and statistical NLU were entirely separate research paradigms, a situation memorably caricatured by Gazdar (1996). In the early 1990s, NLU rapidly 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 bringing increasing amounts of linguistic knowledge into statistical techniques (Charniak *et al.*, 2003; Huang *et al.*, 2006), and this trend looks set to continue[14]. The lesson from NLU and MT appears to be that higher quality results when the symbolic and statistical paradigms join forces.

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 intermediate non-language representations of content. If NLG is to make a comeback 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 was supported by the CoGenT project, a recently completed research project funded under UK EPSRC Grant GR/S24480/01. Many thanks to John Carroll, Roger Evans, Gerald Gazdar, Daniel Paiva, Ehud

[14] The NAACL'07 Workshop on Syntax and Structure in Statistical Translation is also a sign of the times.

Reiter, Kees van Deemter, David Weir and in particular the anonymous NLE reviewers, for very helpful comments. Also gratefully acknowledged is the contribution of the (anonymous) ENLG'05 reviewer who suggested the basic idea behind greedy roulette-wheel selection.

References

Bateman, J. (1987) Enabling technology for multilingual natural language generation: the KPML development environment. *Natural Language Engineering*, **3**(1):15–55.

Belz, A. (2004) Context-free representational underspecification for NLG. Technical Report ITRI-04-08, Information Technology Research Institute, University of Brighton.

Belz, A. (2005) Statistical generation: Three methods compared and evaluated. *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG'05)*, pp. 15–23.

Belz, A. (2005b) Corpus-driven generation of weather forecasts. *Proceedings of the 3rd Corpus Linguistics Conference (CL'05)*, http://www.corpus.bham.ac.uk/PCLC.

Belz, A. (2006) *pCRU: Probabilistic generation using representational underspecification.* Technical Report ITRI-06-01, NLTG, CMIS, University of Brighton.

Belz, A. and Reiter, E. (2006) Comparing automatic and human evaluation in NLG. *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*, pp. 313–320.

Bouayad-Aga, N., Scott, D. and Power, R. (2000) Integrating Content and Style in Documents: A case study of Patient Information Leaflets. *Information Design Journal*, **9**(2–3):161–176.

Briscoe, T. and Carroll, J. (1992) Probabilistic normalisation and unpacking of packed parse forests for unification-based grammars. *Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pp. 33–38.

Briscoe, T. and Carroll, J. (1993) Generalised probabilistic LR parsing of natural language (corpora) with unification-based grammars. *Computational Linguistics*, **19**(1):25–59.

Cahill, L., Doran, C., Evans, R., Mellish, C., Paiva, D., Reape M. and Scott, D. (1999) In Search of a Reference Architecture for NLG Systems. *Proceedings of the 7th European Workshop on Natural Language Generation (ENLG'99)*, pp. 77–85.

Charniak E. (2000) A maximum-entropy-inspired parser. *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL'00)*, pp. 132–139.

Charniak, E., Knight, K. and Yamada, K. (2003) Syntax-based language models for machine translation. *Proceedings of the 9th Machine Translation Summit (MT-Summit IX)*, pp. 40–46.

Collins, M. (2000) Discriminative reranking for natural language parsing. *Proceedings of ICML 2000*, pp. 175–182.

Doddington, G. (2002) Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. *Proceedings of the ARPA Workshop on Human Language Technology*, pp. 128–132.

Elhadad, M. and Robin, J. (1996) *An Overview of SURGE: A reusable comprehensive syntactic realization component.* Technical Report 96-03, Dept of Mathematics and Computer Science, Ben Gurion University, Beer Sheva, Israel.

Gazdar, G. (1996) Paradigm merger in natural language processing. In: Milner, R. and Wand, I. (eds), *Computing Tomorrow: Future Research Directions in Computer Science*, pp. 88–109. Cambridge University Press.

Goldberg, E., Driedger, N. and Kittredge, R. (1994) Using natural-language processing to produce weather forecasts. *IEEE Expert*, **9**(2):45–53.

Habash, N. (2004) The use of a structural n-gram language model in generation-heavy hybrid machine translation. In: Belz, A., Evans, R. and Piwek, P. (eds), *Proceedings of the Third International Conference on Natural Language Generation (INLG'04)*, volume 3123 of *LNAI*, pp. 61–69. Springer.

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

Huang, L., Knight, K. and Joshi, A. (2006) Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th biennial conference of the Association for Machine Translation in the Americas (AMTA'06)*, pp. 66–73.

Humphreys, K., Calcagno, M. and Weise, D. (2001) Reusing a statistical language model for generation. *Proceedings of the 8th European Workshop on Natural Language Generation (ENLG'01)*, pp. 86–91.

Isabelle, P. (1984) Machine translation at the TAUM group. In M. King (ed.) *Machine Translation Today: The State of the Art*, Edinburgh University Press.

Kasper, R. (1989) A flexible interface for linking applications to PENMAN's sentence generator. *Proceedings of the HLT'89 Workshop on Speech and Natural Language*, pp. 153–158.

Knight, K. and Langkilde, I. (1998) Generation that exploits corpus-based statistical knowledge. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING-ACL'98)*, pp. 704–710.

Langkilde-Geary, I. (2005) An Exploratory Application of Constraint Optimization in Mozart to Probabilistic Natural Language Processing. *Proceedings of the International Workshop on Constraint Solving and Language Processing (CSLP)*, LNAI, vol. 3438, pp. 172–183. Springer.

Lavoie, B. and Rambow, O. (1997) A fast and portable realizer for text generation systems. *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP'97)*, pp. 265–268.

Lin, C.-Y. and Hovy, E. (2003) Automatic evaluation of summaries using n-gram co-occurrence statistics. *Proceedings of HLT-NAACL 2003*, pp. 71–78.

McKeown, K., Kukich, K. and Shaw, J. (1994) Practical issues in automatic documentation generation. *Proceedings of the 3rd Applied Natural Language Processing Conference (ANLP'94)*, pp. 7–14.

Magerman, D. and Marcus, M. (1991) Pearl: A probabilistic chart parser. *Proceedings of the 2nd International Workshop on Parsing Technologies*, pp. 193–199.

Mann, W. and Mathiesen, C. (1983) *NIGEL: A systemic grammar for text generation.* Technical Report ISI/RR-85-105, Information Sciences Institute.

Mann, W. and Mathiesen, C. (1985) Demonstration of the Nigel text generation computer program. In: Benson, R. and Greaves, J. (eds), *Systemic Perspectives on Discourse: Selected Papers from the 9th International Systemics Workshop*, Volume 1, pp. 50–83. Ablex.

Manning, C. and Schuetze, H. (1999) *Foundations of Statistical Natural Language Processing.* The MIT Press.

Marciniak, T. and Strube, M. (2004) Classification-based generation using TAG. In: Belz, A., Evans, R. and Piwek, P. (eds), *Proceedings of the Third International Conference on Natural Language Generation (INLG'04)*, volume 3123 of *LNAI*, pp. 100–109. Springer.

Marciniak, T. and Strube, M. (2005) Discrete optimization as an alternative to sequential processing in NLG. *Proceedings of 10th European Workshop On Natural Language Generation (ENLG'05)*, pp. 101–108.

Mellish, C., Reape, M., Scott, D., Cahill, L., Evans, R. and Paiva, D. (2006) A Reference Architecture for Natural Language Generation Systems. *Natural Language Engineering*, **12**(1):1–34. (Corrected version; originally published: 2004, In: Cunningham, H. and Scott, D. (eds), Special Issue on Software Architecture for Language Engineering, pp. 227–260.)

Oh, A. and Rudnicky, A. (2000) Stochastic language generation for spoken dialogue systems. *Proceedings of the ANLP-NAACL 2000 Workshop on Conversational Systems*, pp. 27–32.

Paiva, D. and Evans, R. (2005) Empirically based control of natural language generation. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 58–65.

Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J. (2001) *BLEU: A method for automatic evaluation of machine translation.* IBM research report, IBM Research Division.

Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J. (2002) BLEU: A method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pp. 311–318.

Power, R. (2000) Planning texts by constraint satisfaction. *Proceedings of the 18th International Conference on Computational Linguistics (COLING'00)*, pp. 642–648.

Ratnaparkhi, A. (2000) Trainable methods for surface natural language generation. *Proceedings of the 6th Applied Natural Language Processing Conference and the 1st Meeting of the North American Chapter of the Association of Computational Linguistics (ANLP-NAACL'00)*, pp. 194–201.

Reiter, E. (1994) Has a consensus NL generation architecture appeared and is it psycholinguistically plausible? *Proceedings of the 7th International Workshop on Natural Language Generation INLG'94*, pp. 163–170.

Reiter, E. and Dale, R. (1997) Building Applied Natural Language Generation Systems. *Natural Language Engineering*, **3**(1):57–87.

Reiter, E. and Dale, R. (2000) *Building Natural Language Generation Systems.* Cambridge University Press.

Reiter, E. and Sripada, Y. (2002) Should corpora texts be gold standards for NLG? *Proceedings of the 2nd International Conference on Natural Language Generation (INLG'02)*, pp. 97–104.

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

Riezler S. and Maxwell III, J. T. (2005) On some pitfalls in automatic evaluation and significance testing for MT. *Proceedings of the ACL'05 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, pp. 57–64.

Sripada, Y., Reiter, E., Hunter, J. and Yu, J. (2002) SumTime-Meteo*: Parallel corpus of naturally occurring forecast texts and weather data.* Technical Report AUCS/TR0201, Computing Science Department, University of Aberdeen.

Stolcke, A. (2002) SRILM: An extensible language modeling toolkit. *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP'02)*, pp. 901–904.

Varges, S. and Mellish, C. (2001) Instance-based natural language generation. *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL'01)*, pp. 1–8.

White, M. (2004) Reining in CCG chart realization. In: Belz, A., Evans, R. and Piwek, P. (eds), *Proceedings of the Third International Conference on Natural Language Generation (INLG'04).* Volume 3123 of *LNAI*, pp. 182–191. Springer.