

Assessing the Trade-Off between System Building Cost and Output Quality in Data-To-Text Generation

Anja Belz and Eric Kow

Natural Language Technology Group
School of Computing, Mathematical and Information Sciences
University of Brighton
Brighton BN2 3PB, UK
<http://www.nltg.brighton.ac.uk/>

Abstract. Data-to-text generation systems tend to be knowledge-based and manually built, which limits their reusability and makes them time and cost-intensive to create and maintain. Methods for automating (part of) the system building process exist, but do such methods risk a loss in output quality? In this paper, we investigate the cost/quality trade-off in generation system building. We compare six data-to-text systems which were created by predominantly automatic techniques against six systems for the same domain which were created by predominantly manual techniques. We evaluate the systems using intrinsic automatic metrics and human quality ratings. We find that there is some correlation between degree of automation in the system-building process and output quality (more automation tending to mean lower evaluation scores). We also find that there are discrepancies between the results of the automatic evaluation metrics and the human-assessed evaluation experiments. We discuss caveats in assessing system-building cost and implications of the discrepancies in automatic and human evaluation.

1 Introduction

Traditional Natural Language Generation (NLG) systems tend to be handcrafted knowledge-based systems. Such systems tend to be brittle, expensive to create and hard to adapt to new domains or applications. Over the last decade or so, in particular following Knight and Langkilde's work on n-gram-based generate-and-select surface realisation [13, 16], NLG researchers have become increasingly interested in systems that are automatically trainable from data. Systems that have a trainable component tend to be easier to adapt to new domains and applications, and increased automation in system building is often taken as self-evidently a good thing. The question is, however, whether reduced system building cost and increased adaptability are achieved at the price of a reduction in output quality, and if so, how great the price is. This in turn raises the question of how to evaluate output quality so that a potential decrease can be detected and quantified.

In this paper we set about trying to find answers to these questions. We start, in the following section, by briefly describing the corpus of weather forecasts which we used in our experiments. In the next section (Section 3), we outline four different approaches to building data-to-text generation systems which involve different combinations of manual and automatic techniques. Next (Section 4) we describe ten systems in

the four categories which we used in the original set of experiments. In Section 5 we describe the human-assessed and automatically computed evaluation methods we used to comparatively evaluate the quality of the outputs of the ten systems. We then present the evaluation results and discuss implications of discrepancies we found between the results of the human and automatic evaluations (Section 6).

In material not previously reported in Belz & Kow 2009 [5], we also present the results of some follow-up experiments involving two additional, newly built systems, which we carried out in order to look into the impact of alternative input representations on the fully automatically trainable systems (Section 7), and present some further discussion of (i) issues in assessing the extent to which a system has been built manually, and (ii) the implications of the discrepancies we see between automatic and human-assessed evaluations (Section 8).

2 Data

The main component of the Prodigy-METEO corpus is a set of pairs of wind data (input) and corresponding wind forecast text (output).¹ The Prodigy-METEO outputs were extracted from the SumTime-METEO corpus [26]. We used only those forecasts from SumTime-METEO that are for the period 06:00–24:00 GMT. These are issued in the a.m. and make up roughly half of SumTime-METEO. An extract from an example forecast file (for 5Oct2000_03) is shown in Figure 2. From this subset of forecasts, we then extracted all ‘wind statements’ except the one for the long range outlook (i.e. all wind forecasts statements under points 2, 3 and 4, for 10m and 50m. Only the first wind statement from each a.m. forecast was included in Prodigy-METEO (for full details of the corpus creation process, see the corpus release notes [2]).

The Prodigy-METEO inputs (an example of such an input can be found at the top of Table 1) are vectors of time stamps and wind parameters, and were ‘reverse-engineered’, by automatically aligning wind speeds and wind directions in the forecasts with time-stamps in the wind data file (an extract from the wind data file for 5 Oct 2000 is shown in Figure 1). 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 entirely 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.² From each of these alignments, numerical data vectors were automatically created; e.g. the example at the top of Table 1 is the input vector for 5Oct2000_03. The input vector is a sequence of 7-tuples $\langle i, d, s_{min}, s_{max}, g_{min}, g_{max}, t \rangle$

¹ In response to requests for our data, we have now released a pre-alpha version of the Prodigy-METEO Corpus which also includes outputs from the 10 systems described in Section 4 that were built with this data, as well as some additional human-authored wind forecasts. The corpus can be downloaded here: <http://www.nltg.brighton.ac.uk/home/Anja.Belz/> For full details of the corpus contents, see Belz, 2009b [2].

² 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.

```

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
...

```

Fig. 1. Meteorological data file for 05-10-2000, a.m. (names of oil fields 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. 2. Wind forecast for 05-10-2000, a.m. (names of oil fields anonymised).

where i is the tuple's ID, d is the wind direction, s_{min} and s_{max} are the minimum and maximum wind speeds, g_{min} and g_{max} are the minimum and maximum gust speeds, and t is a time stamp (indicating for what time of the day the data is valid).

In order to obtain these input vectors, we chunked the forecast texts into adjacent phrases, each of which realises one 7-tuple. Each forecast corresponds to at least one 7-tuple. One or more parts of a 7-tuple may not be realised in a given forecast. A -1 value for a timestamp t means that the procedure described above failed to identify a time for a segment. A '-' value means that the corresponding wind information is not included in the forecast text.

The Prodigy-METEO corpus consists of 483 wind data/forecast pairs. For the purposes of the experiments reported in this and previous papers we created 5 randomly determined subdivisions of the corpus into training and test data, for 5-fold cross-validation. For all generation methods that involve training from data, we repeated the training and testing process 5 times, once for each fold, as is standard under cross-validation. For the automatically computed evaluation methods, we therefore compute scores for each of the 5 test sets, and then average over them. In order to be able to compare the forecasts produced by our systems directly to those produced by the original SUMTIME system,³ we had to remove a small number of forecasts from each test set for which we did not have SumTime system outputs.

In the human-assessed evaluation experiment (see Section 5.2), where we used 22 randomly selected forecast dates, we needed exactly one output from each system for each of the 22 dates. For the trainable systems we therefore randomly selected a fold for each date to obtain an output from.

³ The SUMTIME-Hybrid system to be precise (see Section 4.1).

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
SUMTIME-Hybrid	SSW 16-20 GRADUALLY BACKING SSE THEN BECOMING VARIABLE 10 OR LESS BY MIDNIGHT
PCFG-greedy	SSW 16-20 BACKING SSE FOR A TIME THEN FALLING VARIABLE 4-8 BY LATE EVENING
PCFG-roulette	SSW 16-20 GRADUALLY BACKING SSE AND VARIABLE 4-8
PCFG-viterbi	SSW 16-20 BACKING SSE VARIABLE 4-8 LATER
PCFG-2gram	SSW 16-20 BACKING SSE VARIABLE 4-8 LATER
PCFG-random	SSW 16-20 AT FIRST FROM MIDDAY BECOMING SSE DURING THE AFTERNOON THEN VARIABLE 4-8
PSCFG-semantic	SSW 16-20 BACKING SSE THEN FALLING VARIABLE 04-08 BY LATE EVENING
PSCFG-unstructured	SSW 16-20 GRADUALLY BACKING SSE THEN FALLING VARIABLE 04-08 BY LATE EVENING
PBSMT-unstructured	LESS SSW 16-20 SOON BACKING SSE BY END OF THEN FALLING VARIABLE 04-08 BY LATE EVENING
PBSMT-structured	GUSTS SSW 16-20 BY EVENING STEADILY LESS GUSTS GRADUALLY BACKING SSE BY LATE EVENING MINONE BY MIDDAY THEN AND FALLING UNKNOWN VARIABLE 04-08 LATER GUSTS

Table 1. Example input with corresponding outputs by all systems and from the corpus (for 5 Oct 2000).

3 Four Ways to Build an NLG Systems

In this section, we describe four approaches to building language generators involving different combinations of automatic and manual techniques: traditional handcrafted systems (Section 3.1); handcrafted but trainable probabilistic context-free grammar (PCFG) generators (Section 3.2); partly automatically constructed and trainable probabilistic synchronous context-free grammar (PSCFG) generators; and generators automatically built with phrase-based statistical machine translation (PBSMT) methods (Section 3.4). In Section 4 we explain how we used these techniques to build the systems in our evaluation.

3.1 Rule-based NLG

Traditional NLG systems are handcrafted as rule-based 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 analysis and expert consultation. Reiter and Dale’s influential 1997 paper [23] recommended that NLG systems be built largely “by careful analysis of the target text corpus, and by talking to domain experts” (p. 74, and reiterated on pp. 58, 61, 72 and 73).

Handcrafted generation tools have always formed the mainstay of NLG research, a situation virtually unchanged by the statistical revolution that swept through other NLP fields in the 1990s. Well-known examples include the surface realisers Penman,

FUF/SURGE and RealPro, the referring expression generation components created by Dale and Reiter, and content-to-text generators built in the M-PIRO and PLAN-Doc projects, to name but a very few.⁴

3.2 PCFG generation

Context-free grammars (CFG) are non-directional, and can be used for generation as well as for analysis (parsing). One approach that uses CFGs for generation is Probabilistic Context-free Representationally Underspecified (*p*CRU) language generation [1]. As mentioned above, traditional NLG systems tend to be composed of generation rules that apply transformations to representations. 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.

In this approach, a CFG is created by hand that encodes the space of all possible generation processes from inputs to outputs, and has no decision-making ability. A probability distribution over this base CFG is estimated from a corpus, and this is what enables decisions between alternative generation rules to be made. The *p*CRU package permits this distribution to be used in one of the following three modes to drive generation processes: (i) greedy – apply only the most likely rule at each choice point; (ii) Viterbi – apply all expansion rules to each nonterminal to create the generation forest for the input, then do a Viterbi search of the generation forest; (iii) greedy roulette-wheel – select a rule to expand a nonterminal according to a non-uniform random distribution proportional to the likelihoods of expansion rules.

In addition there are two baseline modes: (i) random – where generation rules are randomly selected at each choice point; and (ii) n-gram – where all alternatives are generated and the most likely is selected according to an n-gram language model (as in NITROGEN [13] and its successor HALOGEN [16]).

For the linguistically constrained weather forecasting domain, *p*CRU generators trained on raw corpora have been shown to perform well [1], but for more complex domains it is likely that manually annotated corpora will be needed for training the CFG base generator. As this is in addition to the manually constructed CFG base generator, the manual component in PCFG generator building is potentially substantial.

3.3 PS CFG generation

Synchronous context-free grammars (SCFGs) are mostly used in machine translation [10], but have also been used for simple content-to-text generation [28]. The simplest form of SCFG can be viewed as a pair of CFGs G_1, G_2 with paired production rules such that for each rule in G_1 there is a rule in G_2 with the same left-hand side, and the same non-terminals in the right-hand side (RHS). The order of non-terminals on the RHS may differ, and each RHS may additionally contain any number of terminals in any order. An SCFG can equivalently be seen as a single grammar G encoding a set of pairs of

⁴ See <http://www.nlg-wiki.org/> for information about all these systems and their creators.

strings. A probabilistic SCFG is defined by the 6-tuple $G = \langle \mathcal{N}, \mathcal{T}_e, \mathcal{T}_f, L, S, \lambda \rangle$, where \mathcal{N} is a finite set of non-terminals, $\mathcal{T}_e, \mathcal{T}_f$ are finite sets of terminal symbols, L is a set of paired production rules, S is a start symbol $\in \mathcal{N}$, and λ is a set of parameters that define a probability distribution of derivations under G . Each rule in L has the form $A \rightarrow \langle \alpha; \beta \rangle$, where $A \in \mathcal{N}$, $\alpha \in N \cup \mathcal{T}_e^+$, $\beta \in N \cup \mathcal{T}_f^+$, and $N \subseteq \mathcal{N}$. SCFGs can be trained from aligned corpora to produce probabilistic (or ‘weighted’) SCFGs.

In MT the two CFGs that make up an SCFG are used to encode (the structure of) the two languages which the MT system translates between. Translation with an SCFG then consists of (i) parsing the input string with the source language CFG to produce a derivation tree, and then (ii) generating along the same derivation tree, but using the target language CFG to produce the output string.

When using SCFGs for content-to-text generation one of the paired CFGs encodes the meaning representation language, and the other the (natural) language in which text is supposed to be generated. A generation process then consists of (i) ‘parsing’ the meaning representation (MR) into its constituent structure, and, in the opposite direction, (ii) assembling strings of words corresponding to constituent parts of the input MR into a sentence or text that realises the entire MR.

We used the $WASP^{-1}$ method [27, 28] which provides a way in which a probabilistic SCFG can be constructed for the most part automatically. The training process requires two resources as input: a CFG of MRs and a set of sentences paired with their MRs. As output, it produces a probabilistic SCFG. The training process works in two phases, producing a (non-probabilistic) SCFG in the *lexical acquisition phase*, and associating the rules with probabilities in the *parameter estimation phase*.

The lexical acquisition phase uses the GIZA++ word-alignment tool, an implementation [17] of IBM Model 5 [8] to construct an alignment of MRs with NL strings. An SCFG is then constructed by using the MR CFG as a skeleton and inferring the NL grammar from the alignment.

For the parameter estimation phase, $WASP^{-1}$ uses a log-linear model from Koehn et al. [15] which defines a conditional probability distribution over derivations D given an input MR f as

$$P_\lambda(D|f) \propto P(e(D))^{\lambda_1} \prod_{d \in D} w_\lambda(r(d))$$

where $e(D)$ is the output sentence that the derivation D yields and $w_\lambda(r(d))$ is the weight an individual rule used in a derivation, defined as

$$w_\lambda(A \rightarrow \langle \alpha, \beta \rangle) = P(\beta|\alpha)^{\lambda_2} P(\alpha|\beta)^{\lambda_3} P_w(\beta|\alpha)^{\lambda_4} P_w(\alpha|\beta)^{\lambda_5} \exp(-|\alpha|)^{\lambda_6}$$

where $P(\beta|\alpha)$ and $P(\alpha|\beta)$ are the relative frequencies of β and α , $P_w(\beta|\alpha)$ and $P_w(\alpha|\beta)$ are lexical weights, and $\exp(-|\alpha|)$ is a word penalty to control output sentence length. The model parameters λ_i are trained using minimum error rate training.

Compared to probabilistic CFGs, probabilistic SCFGs trained with $WASP^{-1}$ have a much reduced manual component in system building. In the latter, the NL grammar for the output language, the mapping from MRs to word strings and the rule probabilities

are all created automatically, moreover from raw corpora, whereas in PCFGs, only the rule probabilities are obtained automatically.

3.4 SMT methods

A Statistical Machine Translation (SMT) system is essentially composed of a translation model and a language model, where the former translates source language substrings into target language substrings, and the language model determines the most likely linearisation of the translated substrings. The currently most popular phrase-based SMT (PBSMT) approach translates phrases (arbitrary sequences of words, rather than phrases in the linguistic sense), whereas the original ‘IBM models’ translated words. Different PBSMT methods differ in how they construct the phrase translation table.

We used the phrase-based translation model included in the MOSES toolkit [14] which is based on the noisy channel model, where Bayes’s rule is used to reformulate the task of translating a source language string f into a target language string \hat{e} as finding the sentence \hat{e} such that $\hat{e} = \operatorname{argmax}_e P(e)P(f|e)$. The translation model (which gives $P(f|e)$) is obtained from a parallel corpus of source and target language texts, where the first step is automatic alignment using the GIZA++ word-level aligner. Word-level alignments are used to obtain phrase translation pairs using a set of heuristics. A 3-gram language model (which gives $P(e)$) for the target language is trained either on the same or a different corpus. For full details refer to Koehn et al. [15, 14].

PBSMT offers a completely automatic method for constructing generators from given corpora of paired MRS and realisations, on the basis of which the PBSMT approach constructs a mapping from MRS to realisations.

4 Ten Weather Forecast Text Generators

4.1 SUMTIME-Hybrid

We included the original SUMTIME system [24] in our evaluations. This rule-based system has two modules: a content-determination module and a microplanning and realisation module. It can be run without the content-determination module, taking content representations (7-tuple sequences as described in Section 2) as inputs, and is then called SUMTIME-Hybrid. SUMTIME-Hybrid is a traditional deterministic rule-based generation system. Table 1 shows an example forecast from the SUMTIME-Hybrid system (and corresponding outputs from the other systems, described below).

4.2 PCFG generators

We also included five *p*CRU generators for the METEO domain created previously [1]. The *p*CRU base grammar for the Prodigy-METEO data is a set of generation rules with atomic arguments that convert an input into a set of forecast texts. To create inputs for the *p*CRU generators from the corpus input vectors (Section 2), first information is added (automatically) to each of the 7-tuples encoding whether the change in wind direction compared to the preceding 7-tuple is clockwise or anti-clockwise; whether change in

wind speed is an increase or a decrease; and whether a 7-tuple is the last in the vector. Then, the augmented 7-tuples are converted into nonterminals with arguments.

A probability distribution over the base generator was obtained by the multi-treebanking method [1] from the (un-annotated) Prodigy-METEO corpus. This method first parses the corpus with the base CFG and then obtains rule-application frequency counts from the parsed corpus which are used to obtain a probability distribution by straightforward maximum likelihood estimation. If there is more than one parse for a sentence then the frequency count increment is equally split over rules in alternative parses.

4.3 PSCFG generators

We created two probabilistic synchronous CFG (PSCFG) generators for the METEO domain using $WASP^{-1}$. The main task here was to create a CFG for wind data representations. We used two different grammars (resulting in two different generators). The ‘unstructured’ grammar encodes raw corpus input vectors augmented as described in Section 4.2, whereas the ‘semantic’ grammar encodes representations with recursive predicate-argument structure that more resemble semantic forms. These were produced automatically from the raw input vectors.

Both the PSCFG-unstructured and the PSCFG-semantic generators were built in the same way, by feeding the CFG for wind data representations and the corpus of paired wind data representations and forecasts to $WASP^{-1}$ which then created PSCFGs from it.

4.4 PBSMT generators

We also created two generators with the MOSES toolkit. The main question here was how to represent the ‘source language’ inputs. While SMT methods are often applied with no linguistic knowledge at all (and are therefore blind as to whether paired inputs and outputs are NL strings or something else), it was not clear how well they would cope with the task of mapping from number/symbol vectors to NL strings. We tested two different input representations, one of which was simply the augmented corpus input vectors as described above (PBSMT-unstructured), and another in which the individual 7-tuples of which the vectors are composed are explicitly marked by predicate-argument structure (PBSMT-structured). As in Wong & Mooney’s content-to-text generation work [28] we wanted to test the effect of treating the structure markers as tokens.

We built two different generators by feeding the two different versions of the paired corpus to MOSES. We did not use a factored translation model (the words used in weather forecasts did not vary sufficiently), and we did not tune our parameters to optimise for BLEU (or other) scores, a method common in SMT in [7, 18].

5 Evaluation Methods

5.1 Automatic evaluation methods

The two automatic metrics used in the evaluations, NIST⁵ and BLEU,⁶ have been shown to correlate well with expert judgments (Pearson’s $r = 0.82$ and 0.79 respectively)

⁵ http://cio.nist.gov/esd/emaildir/lists/mt_list/bin00000.bin

⁶ <ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl>

System	NIST	Homogeneous subsets						
		A	B	C	D	E	F	G
corpus	4.062	A						
PCFG-greedy	3.361		B					
PSCFG-semantic	3.303		B					
PSCFG-unstructured	3.191		B	C				
PCFG-roulette	3.033			C	D			
PBSMT-unstructured	2.924				D			
PCFG-viterbi	2.854				D	E		
PCFG-2gram	2.854				D	E		
SUMTIME-Hybrid	2.707					E	F	
PCFG-random	2.540						F	
PBSMT-structured	2.331							G

Table 2. Mean forecast-level NIST scores and homogeneous subsets (Tukey HSD, alpha = .05) for test data.

in the METEO domain [3]. BLEU- x is an n -gram based string comparison measure, originally proposed by Papineni et al. [19] for evaluation of MT systems. It computes the proportion of word n -grams of length x and less that a system output shares with several reference outputs, and ranges from 0 to 1. Setting $x = 4$ (i.e. considering all n -grams of length ≤ 4) is standard. NIST [11] is a version of BLEU, but where BLEU gives equal weight to all n -grams, NIST gives more importance to less frequent (hence more informative) n -grams, and the range of NIST scores depends on the size of the test set. Some research has shown NIST to correlate with human judgments more highly than BLEU [11, 25, 3].

5.2 Human evaluation

We designed an experiment in which participants were asked to rate forecast texts for Clarity and Readability on scales of 1–7. Clarity was explained as indicating how understandable a forecast was, and Readability as indicating how fluent and readable it was. After an introduction and detailed explanations, participants carried out the evaluations over the web. They were able to interrupt and resume the evaluation at any time.

We randomly selected 22 forecast dates and used outputs from the 10 systems described in Section 4 for those dates (as well as the corresponding forecasts in the corpus) in the evaluation, i.e. a total of 242 forecast texts. We used a repeated Latin squares design where each combination of forecast date and system is assigned two trials. As there were 2 evaluation criteria, there were a total of 968 individual ratings in this experiment. An evaluation session started with three training examples; the real trials were then presented in random order.

We recruited 22 participants from among our university colleagues whose first language was English and who had no experience of NLP. We did not try to recruit master mariners as in earlier experiments reported by Belz and Reiter [3], because these experiments also demonstrated that the correlation between the ratings by such expert evaluators and lay-people is very strong in the METEO domain (Pearson’s $r = 0.845$).

System	BLEU	Homogeneous subsets							
		A	B	C	D	E	F	G	H
corpus	1.00	A							
PCFG-greedy	.65		B						
PSCFG-semantic	.637		B						
PSCFG-unstructured	.617		B	C					
PCFG-viterbi	.57			C	D				
PCFG-2gram	.561				D				
PCFG-roulette	.516				D	E			
PBSMT-unstructured	.500					E			
SUMTIME-Hybrid	.437						F		
PBSMT-structured	.338							G	
PCFG-random	.269								H

Table 3. Mean forecast-level BLEU scores and homogeneous subsets (Tukey HSD, alpha = .05) for test data.

6 Results

For each evaluation method, we carried out a one-way ANOVA with ‘System’ as the fixed factor, and the evaluation measure as the dependent variable. In each case we report the main effect of System on the measure and (if it is significant) we also report significant differences between pairs of systems in the form of homogeneous subsets obtained with a post-hoc Tukey HSD analysis.

Tables 2 and 3 display the results for the BLEU and NIST evaluations, where scores were calculated on test data, using 5-fold cross-validation. System names are shown in the first column, mean forecast-level scores in the second, and the remaining columns indicate significant differences between systems. The way to read the homogeneous subsets is that two systems which do not have a letter in common are significantly different with $p < .05$.

For the BLEU evaluation, the main effect of System on BLEU score was $F_{(10,2420)} = 248.274$, at $p < .001$. PCFG-greedy, PSCFG-semantic and PSCFG-unstructured come out top, although only the first two are significantly better than all other systems. SUMTIME-Hybrid, PBSMT-structured and PCFG-random bring up the rear, with the remaining systems distributed over the middle ground. A striking result is that the handcrafted SUMTIME-Hybrid system comes out near the bottom, being significantly worse than all other systems except PCFG-structured and PBSMT-random.

For the NIST evaluation, the main effect of System on BLEU score was $F_{(10,2420)} = 108.086$, at $p < .001$. The systems were ranked in the same way as in the BLEU evaluation except for the systems in the D subset. The correlation between the NIST and BLEU scores is Pearson’s $r = .739$, $p < .001$.

The main results from the automatic evaluations are that the two PSCFG systems and the PCFG system with the greedy generation algorithm are best overall. However, the human evaluations produced rather different results.

Figure 3 is a series of bar charts representing the results of the human evaluation for Clarity. For each system (indicated by the labels on the x-axis), there are 7 bars, showing how many ratings of 1, 2, 3, 4, 5, 6 and 7 (7 being the best) a system was given.

	Scores on data from human evaluation			
	Clarity	Readability	NIST	BLEU
SUMTIME-Hybrid	6.06	6.18	5.71	0.52
PSCFG-semantic	5.79	5.70	6.76	0.65
corpus	5.79	5.93	8.45	1
PCFG-greedy	5.79	5.63	6.73	0.67
PSCFG-unstruc	5.72	5.84	6.61	0.64
PCFG-roulette	5.29	5.56	6.07	0.52
PCFG-2gram	5.29	5.29	5.23	0.52
PCFG-viterbi	4.90	5.34	5.15	0.51
PCFG-random	4.43	4.52	4.52	0.25
PBSMT-unstruc	3.70	3.93	5.38	0.49
PBSMT-struc	2.79	2.77	4.21	0.33

Table 4. Mean Clarity and Readability ratings from human evaluation; NIST and BLEU scores on same 22 forecasts as used in human evaluation.

So the left-most bar for a system shows how many ratings of 1 a system was given, the second bar how many ratings of 2, etc. Systems are shown in descending order of mode (the value of the most frequently assigned rating, e.g. 7 for PSCFG-unstructured on the left, and 1 for PBSMT-structured on the right). The PSCFG-unstructured and SUMTIME-Hybrid systems come out top in this evaluation, with PSCFG-semantic, PCFG-roulette and PCFG-greedy close behind. Conversely, PBSMT-structured clearly came out worst, with no ratings of 7 and a mode of 1 (=completely unclear).

Figure 4 consists of the same kind of bar charts, for the Readability ratings. Here the SUMTIME-Hybrid system is the clear winner, with no ratings of 1 or 2, and 22 ratings of 7 (=excellent, all parts read well). It is closely followed by PSCFG-unstructured, the corpus forecasts and PSCFG-semantic. Again, PBSMT-structured is clearly worst with no ratings of 7, although this time the mode is 3 (=fairly bad).

We also looked at the means of the ratings, and these are shown in the second and third columns of Table 4. The means have to be treated with some caution, because ratings are ordinal data and it is not clear how meaningful it is to compute means. However, it is a simple way of obtaining a system ranking for comparison with the two automatic scores (shown in the remaining two columns of Table 4, computed over the system outputs used in the human evaluation). In terms of means, SUMTIME-Hybrid comes out top for both Clarity and Readability. In Clarity, it is followed by the two PSCFG systems, the corpus files (the only forecasts actually written by humans), and PCFG-greedy which have virtually the same means. For Readability, corpus and PSCFG-unstructured are ahead of PSCFG-semantic and PCFG-greedy (in this order). Bringing up the rear for both Clarity and Readability, as in the NIST evaluations, is PBSMT-structured, with PCFG-random and and PBSMT-unstructured faring somewhat better.

There are some striking differences between the automatic and human evaluations. For one, the human evaluators rank the SUMTIME-Hybrid system very high, whereas both automatic metrics rank it very low, just above PCFG-random and PBSMT-structured. Furthermore, the metrics rank PBSMT-unstructured more highly than the human evaluators, placing it above the SUMTIME-Hybrid system and in the case of NIST, also above

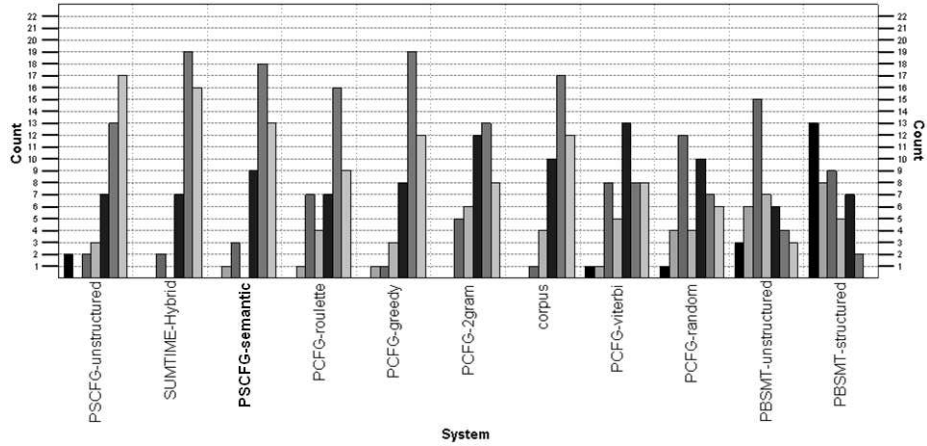


Fig. 3. Clarity ratings: Number of times each system was rated 1, 2, 3, 4, 5, 6, and 7 on Clarity. Systems in descending order of mode (most frequent rating).

two of the PCFG systems (Table 2). In fact, the human and the automatic evaluations agree only that the PSCFG systems and PCFG-greedy are equally good.

7 Issues Around Input Representations

The different input representations required by our systems are all derived from the basic sequence of 7-tuples described in Section 2, augmented by information regarding segment position, changes in wind speed (increase/decrease) and direction (clockwise/anti-clockwise). The exact nature of each system’s input representations, e.g. whether they are structured or flat, is in part determined by system type.

In Figure 5, we give examples of the input representations required by each system. For the PCFG and PSCFG systems input representation is determined entirely by the specific grammar used by the given system (the same is true of the SUMTIME-Hybrid system). As all four PCFG systems use the same base grammar, they all take the same predicate-argument sequences as inputs (see PCFG-* input representation in Figure 5). The two PSCFG systems have different content representation grammars, and their input representations therefore also differ (PSCFG-* in Figure 5).

The PBSMT systems differ from the others in that the entire mapping is automatically constructed, and in principle any input representations could therefore be used in the corpus of paired inputs/outputs that are fed to the system building component (as long as inputs are sequences of tokens). In our previous experiments, we tried out two alternatives (see PBSMT-* in Figure 5). As mentioned above, we first used input representations with structure markers (predicate names, brackets, commas) in place (PBSMT-structured). However, as can be seen from our results (Tables 2, 3 and 4), simply removing these structure markers (as in PBSMT-unstructured) results in a significant improvement in terms of all evaluation methods we used.

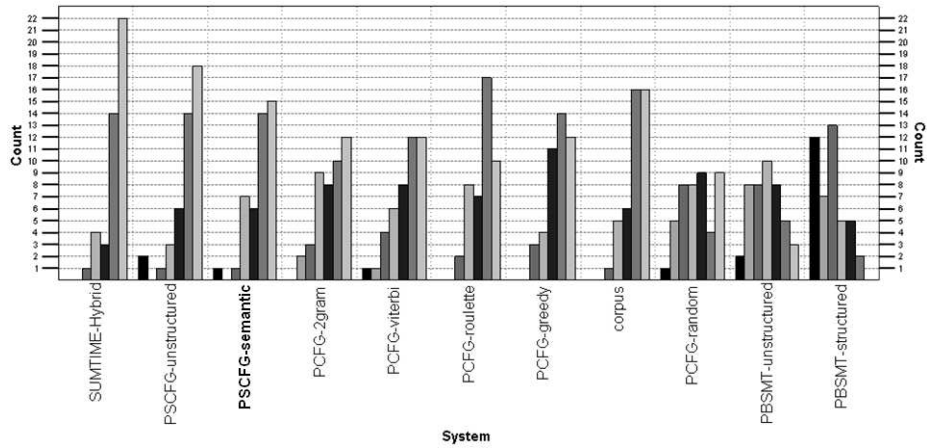


Fig. 4. Readability ratings: Number of times each system was rated 1, 2, 3, 4, 5, 6, and 7 on Readability. Systems in descending order of mode (most frequent rating).

In order to investigate the difference further we created PBSMT-structured-2, a minor variation derived from the PBSMT-structured representation by removing only commas and brackets (leaving the predicate names in).

We also experimented with a rather more different input representation. By looking at the phrase tables used in the other PBSMT generators, we observed that the PBSMT systems often run into problems, because numbers are used with different meanings (e.g. a number can be a segment index or a wind speed), and the phrase tables have no way of distinguishing between one use of the same number and another. The information about which numbers are indices and which are wind speeds is there in the input representations—the first and second elements of each 7-tuple are segment indices, whereas the sixth and seventh elements are wind speeds—but the PBSMT training mechanism has no way of utilising positional information. So for the fourth input representation, PBSMT-tagged, we inserted (in a fully automatic process) the ‘missing’ positional information explicitly in the form of ‘tags’, i.e. short words that precede each relevant token in the input (tags in bold):

PBSMT-tagged	-1 dir _ssw dir2 wind 16 wind2 20 n n six
	windchange same ctrclock dir _sse dir2 wind n wind2 n n n minone
	1 windchange down unknown dir _var wind 04 wind2 08 n n zero

The NIST and BLEU scores for the two new PBSMT systems are shown in Table 5 and 6, respectively, along with scores for the other two PBSMT systems and the corpus texts, for ease of comparison. We are using the same letters to indicate the homogeneous subsets as in Tables 2 and 3.

The results for PBSMT-structured-2 indicate that removing commas and tags leads to the bigger part of the improvement from PBSMT-structured to PBSMT-unstructured, but the latter still outperforms it.

System	Input Representation
Basic input rep.	[[1,SSW,16,20,-,-,0600], [2,SSE,-,-,-,NOTIME],[3,VAR,04,08,-,-,2400]]
PCFG-*	Segment_Init(-1,_SSW,16,20,n,n,six) Segment(2,-1,same,ctrlock,_SSE,n,n,n,minone) Segment(3,1,down,unknown,_VAR,04,08,n,n,zero)
PSCFG-semantic	segment(1,false,nowindchange,wind(dir(ssw,x),16,20),nogust,six) segment(2,false,windchange(same,ctrlock),wind(dir(sse,x),n,n),nogust,minone) segment(3,true,windchange(down,unknown),wind(dir(var,x),04,08),nogust,zero)
PSCFG-unstruct.	-1 _SSW 16 20 n n six 2 -1 same ctrclock _SSE n n n n minone 3 1 down unknown _VAR 04 08 n n zero
PBSMT-structured	segment_init (-1 , _ssw , 16 , 20 , n , n , six) segment (2 , -1 , same , ctrclock , _sse , n , n , n , n , minone) segment (3 , 1 , down , unknown , _var , 04 , 08 , n , n , zero)
PBSMT-unstruct.	-1 _ssw 16 20 n n six 2 -1 same ctrclock _sse n n n n minone 3 1 down unknown _var 04 08 n n zero
Corpus	SSW 16-20 GRADUALLY BACKING SSE THEN FALLING VARIABLE 4-8 BY LATE EVENING

Fig. 5. Example input representations for different generators (for 5 Oct 2000).

System	NIST	Homogeneous subsets			
corpus	4.062	A			
PBSMT-tagged	3.173		B, C		
PBSMT-unstructured	2.924			D	
PBSMT-structured-2	2.831			D	
PBSMT-structured	2.331				G

Table 5. PBSMT systems: Mean forecast-level NIST scores and homogeneous subsets (Tukey HSD, alpha = .05) for test set (cross-validated).

PBSMT-tagged is the best performing of the PBSMT systems; in terms of NIST scores there is no significant difference between it and the top-performing system (PCFG-greedy). This shows that in the case of data-to-text generation systems built using PBSMT methodology, very substantial improvements can be achieved by rewriting input representations. Of course, such improvements are achieved by additional manual work—error analysis and redesigning representations—and we cannot claim that the PBSMT-tagged system was created entirely by automatic techniques. although this took no more than three hours, and if the new tagging technique can be usefully applied to other generation tasks, we can treat the cost as one-off.

8 Discussion

In Section 3 we effectively ranked our four system-building methods in terms of how much manual effort they involve, which gave us the following ranking, in decreasing order: SUMTIME, PCFG-*, PSCFG-*, PBSMT-*. Not one of our four evaluation methods

System	BLEU	Homogeneous subsets			
corpus	1.00	A			
PBSMT-tagged	.576		C, D		
PBSMT-unstructured	.500			E	
PBSMT-structured-2	.481			E	
PBSMT-structured	.338				G

Table 6. PBSMT systems: Mean forecast-level BLEU scores and homogeneous subsets (Tukey HSD, alpha = .05) for test set (cross-validated).

produced exactly the same rank order for these four groups (not if we base the comparison on the best system score in each group nor if we base it on the average), i.e. we could not have predicted output quality rankings with complete accuracy from the degree of automation of the system-building process. Nevertheless, while there is no significant correlation between degree of automation and either BLEU or NIST, there is a significant correlation between degree of automation and both Clarity and Readability.

To establish this, we ranked our original 10 systems in order of degree of automation as follows: SUMTIME, {PCFG-greedy, PCFG-viterbi, PCFG-roulette}, {PCFG-2gram, PCFG-random}, {PSCFG-semantic, PSCFG-unstructured}, PBSMT-unstructured, PBSMT-structured (curly brackets indicating joint rank). We then computed Spearman’s Rho between the automation ranks and the system-level Readability and Clarity scores, resulting in $\rho = -.59$, Sig. (1-tailed) = .036 for Clarity, and $\rho = -.549$, Sig. (1-tailed) = .05 for Readability.⁷ Rho values for BLEU and NIST were close to 0, and not significant.

Our main finding in this research is thus that degree of automation in system building can (to some extent) predict human-assessed intrinsic scores, but automatic metric scores cannot be predicted with it. This conclusion relies on the estimation of degree of automation being accurate; it also raises the question of how to interpret the discrepancy between human-assessed and metric scores. Below we look at each of these issues in slightly more detail.

Assessing System Building Cost: In order to arrive at our estimation of degree of automation, we made the following assumptions: cost was defined as person-months spent in system building from the point at which the system-builder decided to build a wind forecast text generator in the METEO domain up to the point at which the system existed in the form in which we evaluated it; resources available to the system builder from the beginning of this process were the METEO corpus in either the SUMTIME or the PRODIGY version, and *any existing software*. Given these assumptions, the SUMTIME-Hybrid system took on the order of a year to build,⁸ the PCFG systems on the order of 1 month, the PSCFG systems 1 day, and the PBSMT systems 1 hour; within some of the groups we were able to draw finer distinctions (e.g. the PCFG-random and

⁷ We are using 1-tailed significance tests here, because we started out with a 1-tailed (or directional) hypothesis, namely that increasing automation would lead to decreasing scores, and we were testing the strength of the correlation, rather than the direction.

⁸ Belz [1], estimated on the basis of personal communication with E. Reiter and S. Sripada.

PCFG-ngram systems use off-the-shelf tools for selection). None of the systems used existing, reusable generator modules.⁹

The above system ranking is thus appropriate and accurate in our case. In the more general case, however, it is not necessarily the case that handcrafted systems rank highest and PBSMT systems lowest in terms of system-building cost (as defined in the preceding paragraph). For example, not all parts of a completely manually constructed system will have necessarily been built for a given system (i.e. they might have been reused), while use of a fully automatic system-building method can hide substantial manual effort. The cost of building handcrafted systems could very well be reduced with a good set of reusable libraries, or a well designed domain-specific programming language (where “domain” here refers to “natural language generation”). Handcrafted systems could also be partly expressed in terms of some formal grammar which in turn can be made cheaper through judicious use of automation (for example, using a meta-grammar compiler such as XMG [20]). On the other hand, as we saw in Section 7, it is possible to invest manual effort in improving fully automatic system-building methods, in order to try and improve their evaluation scores. In our case this was no more than a few hours, but clearly this could be an open-ended process.

The main point here is that system-building cost must be established on a case-by-case basis, not generically on the basis, say, of system type or generation methodology. While manual effort may well be amortised in the future because (part of) a system is reusable, or, conversely, maintenance costs may accrue in the future, these do not fall under the building-cost heading.

Implications for Evaluation Methods: Given the discrepancy between our human-assessed and automatic evaluation methods, one might wonder which are ‘more right’? BLEU in particular has come in for some hefty criticism over recent years, with some papers (e.g. [9]) exposing individual examples of high BLEU scores for patently bad-quality outputs, and vice versa. We can identify such examples in our data too, e.g.:

System	Output	BLEU
corpus	N-NNE 16-20 RISING 20-24 FOR A TIME THIS AFTERNOON	1
SUMTIME-Hybrid	N-NNE 16-20 INCREASING 20-24 BY MID AFTERNOON	0.43
PCFG	N-NNE 16-20 INCREASING 20-24 FOR A TIME MID PERIOD	0.65

There is really nothing wrong with the above SUMTIME-Hybrid output, in fact there is evidence that master mariners prefer more precise statements of time [22], but its BLEU score is low (whereas the human evaluators give it clear 7s).

Yet human evaluation does not always produce ideal scores either, something that is perhaps not discussed frequently enough (but see e.g. [4]). Human evaluators are notorious for their lack of agreement with other evaluators and even themselves [21]; and individual scores can be just as unintuitive as in the case of BLEU. Again, we can find examples in our data (scores from two evaluators for each of the two forecasts):

⁹ The systems only used reusable interpreters and compilers in addition to the training tools.

System	Output	Clar.	Read.
PCFG-2gram	SSW 26 - 30 VEERING SW 32 - 36 VEERING WSW 20 - 24	3	2
	BACKING SW 28 - 32	5	6
PBSMT-unstruct.	OR LESS S'LY 05 - 10 THIS EVENING INCREASING	7	7
	10 - 14 BY LATE EVENING	1	1

In the first example above, the high readability score by the second evaluator does not seem justified, because of the repetition of *veering* and the absence of ‘link words’ between segments such as *then* and *and*; in the second example, clearly the two evaluators who gave these scores could not disagree more, and again it is the high scores that are unjustified—a forecast cannot start with *or less*.

Both automatic metrics and human evaluators are fallible in individual cases. What matters more is how they perform on the whole, on a set of outputs. If inappropriate scores are noise that disappears ‘in the statistical wash’, then all is well, otherwise (and only then) there is a problem.

Without a third, objective point of reference, we cannot be sure which of our evaluation methods is ‘more right’. What we have found again and again in our evaluation experiments (see also Gatt & Belz [12], and Belz et al. [6], both in this volume), is that different types of evaluation do not necessarily agree with each other, and that we should not regard any single one of them as an objective measure of quality, but rather as assessing one particular aspect of systems. If we want our wind forecasts to be similar to the corpus forecasts, then BLEU and NIST can probably give us a fair indication of that type of similarity; if we are interested in how readable and clear human readers (think they) find our forecasts, then we should look at the Clarity and Readability scores.

9 Conclusions

Reports of research on automating (part of) system building often take it as read that such automation is a good thing. The resulting systems are not often compared to handcrafted alternatives in terms of output quality or other quality criteria, and little is therefore known about the loss of system quality that results from automation. The existence of several independently developed systems for the METEO domain of weather forecasts, to which we have added six new systems in the research reported in this paper, provides a unique opportunity to examine the system building cost vs. system quality trade-off in data-to-text generation.

We investigated 12 systems which fall into four categories in terms of the manual work involved in creating them, ranging from completely manual to completely automatic system building. We used two automatically assessed corpus-similarity metrics and two human assessed quality criteria to evaluate systems. We found some significant correlation between degree of automation of system building and the human-assessed Clarity and Readability criteria, but no correlation between the former and either of the automatic metrics.

We found striking differences between the results from tests of human acceptability and measurements of corpus similarity. Relative to the human ratings, the automatic metrics underestimated the quality of the handcrafted SUMTIME-Hybrid system, but

overestimated the quality of the automatically constructed SMT systems. This will not come as a surprise to those familiar with the machine translation evaluation literature where this is a major complaint about BLEU [9]. From our research (see also e.g. [22]) it seems clear that when the quality of diverse types of systems is compared, automatic metrics such as BLEU on their own do not give a complete and reliable picture, and carrying out additional evaluations is crucial.

Increased reusability and adaptability of systems and components have cost and time benefits, and methods for automatically training systems from data offer advantages in both these respects. However, careful evaluation is needed to ensure that these advantages are not achieved at the price of a reduction in system quality that renders systems unacceptable to human users.

Acknowledgments

The research reported in this paper was supported under EPSRC grant EP/E029116/1 (the Prodigy Project). We thank Emiel Krahmer and the anonymous ENLG'09 reviewers for their helpful comments.

References

1. Belz, A.: Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering* 14(4), 431–455 (2008)
2. Belz, A.: Prodigy-METEO: Pre-alpha release notes (Nov 2009). Tech. Rep. NLTG-09-01, Natural Language Technology Group, CMIS, University of Brighton (2009)
3. Belz, A., Reiter, E.: Comparing automatic and human evaluation of NLG systems. In: *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'06)*. pp. 313–320 (2006)
4. Belz, A.: That's nice ... what can you do with it? *Computational Linguistics* 35(1), 111–118 (2009)
5. Belz, A., Kow, E.: System building cost vs. output quality in data-to-text generation. In: *Proceedings of the 12th European Workshop on Natural Language Generation* (2009)
6. Belz, A., Kow, E., Viethen, J., Gatt, A.: Referring expression generation in context: The GREC shared task evaluation challenges. In: Krahmer, E., Theune, M. (eds.) *Empirical Methods in Natural Language Generation, LNCS*, vol. 5980. Springer, Berlin / Heidelberg (2010)
7. Bertoldi, N., Haddow, B., Fouet, J.: Improved Minimum Error Rate Training in Moses. *The Prague Bulletin of Mathematical Linguistics* 91, 7–16 (2009)
8. Brown, P.F., Della Pietra, V.J., Della Pietra, S.A., Mercer, R.L.: The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics* 19(2), 263–311 (1993)
9. Callison-Burch, C., Osborne, M., Koehn, P.: Re-evaluating the role of BLEU in machine translation research. In: *Proceedings of EACL'06* (2006), citeseer.ist.psu.edu/callison-burch06reevaluating.html
10. Chiang, D.: An introduction to synchronous grammars (part of the course materials for the ACL'06 tutorial on synchronous grammars) (2006)
11. Doddington, G.: Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In: *Proceedings of the ARPA Workshop on Human Language Technology* (2002)

12. Gatt, A., Belz, A.: Introducing shared task evaluation to NLG: The TUNA shared task evaluation challenges. In: Krahmer, E., Theune, M. (eds.) *Empirical Methods in Natural Language Generation*, LNCS, vol. 5980. Springer, Berlin / Heidelberg (2010)
13. Knight, K., Langkilde, I.: Generation that exploits corpus-based statistical knowledge. In: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*. pp. 704–710 (1998)
14. Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E.: Moses: Open source toolkit for statistical machine translation. In: *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*. pp. 177–180 (2007)
15. Koehn, P., Och, F.J., Marcu, D.: Statistical phrase-based translation. In: *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL'03)*. pp. 48–54 (2003)
16. Langkilde, I.: Forest-based statistical sentence generation. In: *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. 170–177 (2000)
17. Och, F.J., Ney, H.: A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics* 29(1), 19–51 (2003)
18. Och, F.: Minimum error rate training in statistical machine translation. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. p. 167. Association for Computational Linguistics (2003)
19. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: A method for automatic evaluation of machine translation. IBM research report, IBM Research Division (2001)
20. Parmentier, Y., Le Roux, J.: XMG: a Multi-formalism Metagrammatical Framework. In: *17th European Summer School in Logic, Language and Information - ESSLLI 2005*. Edinburgh/Scotland (08 2005), <http://hal.inria.fr/inria-00001132/en/>
21. Reidsma, D., Op den Akker, R.: Exploiting ‘subjective’ annotations. In: *Proceedings of the COLING'08 Workshop on Human Judgements in Computational Linguistics*. pp. 8–16 (2008)
22. Reiter, E., Belz, A.: An investigation into the validity of some metrics for automatically evaluating NLG systems. *Computational Linguistics* 35(4) (2009)
23. Reiter, E., Dale, R.: Building applied natural language generation systems. *Natural Language Engineering* 3(1), 57–87 (1997), citeseer.ist.psu.edu/reiter97building.html
24. Reiter, E., Sripada, S., Hunter, J., Yu, J.: Choosing words in computer-generated weather forecasts. *Artificial Intelligence* 167, 137–169 (2005)
25. Riezler, S., Maxwell, J.T.: On some pitfalls in automatic evaluation and significance testing for MT. In: *Proceedings of the ACL'05 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*. pp. 57–64 (2005)
26. Sripada, S., Reiter, E., Hunter, J., Yu, J.: SUMTIME-METEO: A parallel corpus of naturally occurring forecast texts and weather data. Tech. Rep. AUCS/TR0201, Computing Science Department, University of Aberdeen (2002)
27. Wong, Y.W., Mooney, R.: Learning for semantic parsing with statistical machine translation. In: *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL'06)*. pp. 439–446 (2006)

28. Wong, Y.W., Mooney, R.: Generation by inverting a semantic parser that uses statistical machine translation. In: Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL'07). pp. 172–179 (2007)