System Building Cost vs. Output Quality in Data-To-Text Generation

Anja Belz **Eric Kow**
Natural Language Technology Group
University of Brighton
Brighton BN2 4GJ, UK
{asb,eykk10}@bton.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 four new data-to-text systems which were created by predominantly automatic techniques against six existing systems for the same domain which were created by predominantly manual techniques. We evaluate the ten systems using intrinsic automatic metrics and human quality ratings. We find that increasing the degree to which system building is automated does not necessarily result in a reduction in output quality. We find furthermore that standard automatic evaluation metrics underestimate the quality of handcrafted systems and over-estimate the quality of automatically created systems.

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 (Knight and Langkilde, 1998; Langkilde, 2000), 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 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, we briefly describing the SUMTIME corpus of weather forecasts which we used in our experiments. In the next section (Section 2), 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 that generate weather forecast texts in the SUMTIME domain. 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).

2 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 numerical data files paired with a weather forecast. The experiments in this paper focussed on the part of the forecasts that predicts wind characteristics for the next 15 hours.

Figure 1 shows an example data file and Figure 2 shows the corresponding wind forecast written by one of the meteorologists. In Figure 1, the

```
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
...
```

Figure 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
...
```

Figure 2: Wind forecast for 05-10-2000, a.m. (names of oil fields anonymised).

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.

We used a version of the corpus reported previously (Belz, 2008) which contains pairs of wind statements and the wind data that is actually included in the statement, e.g.:

Data: `1 SSW 16 20 - - 0600 2 SSE - - - - NOTIME 3 VAR 04 08 - - 2400`

Text: `SSW 16-20 GRADUALLY BACKING SSE THEN FALLING VARIABLE 4-8 BY LATE EVENING`

The input vector represents a sequence of 7-tuples $\langle i, d, s_{min}, s_{max}, g_{min}, g_{max}, t\rangle$ 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). The corpus consists of 2,123 instances, corresponding to a total of 22,985 words.

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 ten 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 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, 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, Reiter, Horacek and van Deemter, and content-to-text generators built in the PLANDoc and M-PIRO projects, to name but a very few.

3.2 PCFG generation

Context-free grammars 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 (Belz, 2008). 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 $relation(arg_1, ...arg_n) \rightarrow relation_1(arg_1, ...arg_p) ... relation_m(arg_1, ...arg_q)$, $m \geq 1, n, p, q \geq 0$, 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.

In this approach, a CFG is created by hand that encodes the space of all possible generation pro-

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).

cesses 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 HALOGEN).

For the simple SUMTIME domain, *p*CRU generators trained on raw corpora have been shown to perform well (Belz, 2008), 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 PSCFG generation

Synchronous context-free grammars (SCFGs) are used in machine translation (Chiang, 2006), but have also been used for simple concept-to-text generation (Wong and Mooney, 2007). 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. The order of non-terminals on the RHSs may differ, and each RHS may additionally contain any terminals in any order. SCFGs can be trained from aligned corpora to produce probabilistic (or 'weighted') SCFGs.

An SCFG can equivalently be seen as a single grammar G encoding a set of pairs of 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 $N \subseteq \mathcal{N}$.

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 in (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 (Wong and Mooney, 2006; Wong and Mooney, 2007) 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 (Och and Ney, 2003) of IBM Model 5 (Brown et al., 1993) 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. (2003) which defines a conditional probability distribution over derivations d given an input MR f as

$$\Pr_{\lambda}(\mathbf{d}|\mathbf{f}) \propto \Pr(e(d))^{\lambda_1} \prod_{d \in \mathbf{d}} w\lambda(r(d))$$

where $w_\lambda(r(d))$ is the weight an individual rule used in a derivation, defined as

$$w_\lambda(A \rightarrow \langle e, f \rangle) =$$

$$P(f|e)^{\lambda_2} P(e|f)^{\lambda_3} P_w(f|e)^{\lambda_4} P_w(e|f)^{\lambda_5} \exp(-|\alpha|)^{\lambda_6}$$

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, WASP^{-1}-trained probabilistic SCFGs 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 created 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 (an arbitrary sequence of words, rather than 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 proposed by Koehn et al. (2003) and implemented in the MOSES toolkit (Koehn et al., 2007) 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 e as finding the sentence $\mathbf{e}^*$ such that $\mathbf{e}^* = \mathrm{argmax}_{\mathbf{e}} \Pr(\mathbf{e}) \Pr(\mathbf{f}|\mathbf{e})$.

The translation model (which gives $\Pr(\mathbf{f}|\mathbf{e})$) 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 $\Pr(\mathbf{e})$) for the target language is trained either on the same or a different corpus. For full details refer to Koehn et al. (2003; 2007).

PBSMT offers a completely automatic method for constructing generators, where all that is required as input to the system building process is a corpus of paired MRs and realisations, on the basis of which the PBSMT approach constructs a mapping from MSRs to realisations.

4 Ten Weather Forecast Text Generators

4.1 SUMTIME-Hybrid

We included the original SUMTIME system (Reiter et al., 2005) 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 (tuple sequence as described in Section 2) as inputs, and is then called SUMTIME-Hybrid. SUMTIME-Hybrid is a traditional deterministic rule-based generation system, and took about one year to build.[1] Table 1 shows an example forecast from the SUMTIME system (and corresponding outputs from the other systems, described below).

[1]Belz (2008), estimated on the basis of personal communication with E. Reiter and S. Sripada.

4.2 PCFG generators

We also included five *p*CRU generators for the SUMTIME domain created previously (Belz, 2008). The *p*CRU base generator for SUMTIME is a set of generation rules with atomic arguments that convert an input into a set of NL forecasts. To create inputs to the *p*CRU generators, the input vectors as they appear in the corpus (see Section 2) are augmented and converted into sequence of nonterminals: First, information is added to each of the 7-tuples in an automatic preprocessing phase encoding whether the change in wind direction compared to the preceding 7-tuple was clockwise or anti-clockwise; whether change in wind speed was an increase or a decrease; and whether a 7-tuple was the last in the vector. Then, the augmented tuples are converted into a representation of nonterminals with 7 arguments.

A probability distribution over the base generator was obtained by the multi-treebanking method (Belz, 2008) from the un-annotated SUMTIME 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 straighforward 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 SUMTIME 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 probabilistic SCFGs from it.

System	BLEU	Homogeneous subsets							
corpus	1.00	A							
PCFG-greedy	.65		B						
PSCFG-sem	.637		B						
PSCFG-unstr	.617		B	C					
PCFG-viterbi	.57			C	D				
PCFG-2gram	.561				D				
PCFG-roule	.516				D	E			
PBSMT-unstr	.500					E			
SUMTIME	.437						F		
PBSMT-struc	.338							G	
PCFG-rand	.269								H

Table 2: Mean forecast-level BLEU scores and homogeneous subsets (Tukey HSD, alpha = .05) for SUMTIME test sets.

4.4 PBSMT generators

We also created two SUMTIME 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). For comparability with Wong & Mooney (2007) the structure markers were treated 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), or tuning.

5 Evaluation Methods

5.1 Automatic evaluation methods

The two automatic metrics used in the evaluations, NIST[2] and BLEU[3], have been shown to correlate well with expert judgments (Pearson's $r = 0.82$ and 0.79 respectively) in the SUMTIME domain (Belz and Reiter, 2006).

[2]http://cio.nist.gov/esd/emaildir/lists/mt_list/bin00000.bin

[3]ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl

BLEU-x is an n-gram based string comparison measure, originally proposed by Papineni et al. (2001) 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. Setting $x = 4$ (i.e. considering all n-grams of length ≤ 4) is standard. NIST (Doddington, 2002) 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 (Doddington, 2002; Riezler and Maxwell, 2005; Belz and Reiter, 2006).

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 all 10 systems 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 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 Reiter and Belz (2006), because these experiments also demonstrated that the correlation between the ratings by such expert evaluators and lay-people is very strong in the SumTime domain (Pearson's $r = 0.845$).

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.

System	NIST	Homogeneous subsets						
corpus	4.062	A						
PCFG-greedy	3.361		B					
PSCFG-sem	3.303		B					
PSCFG-unstr	3.191		B	C				
PCFG-roule	3.033			C	D			
PBSMT-unstr	2.924				D			
PCFG-viterbi	2.854				D	E		
PCFG-2gram	2.854				D	E		
SumTime	2.707					E	F	
PCFG-rand	2.540						F	
PBSMT-struc	2.331							G

Table 3: Mean forecast-level NIST scores and homogeneous subsets (Tukey HSD, alpha = .05) for SumTime test sets.

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 sets, using a 5-fold cross-validation set-up. System names (in abbreviated form) 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 $F = 248.274$, 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 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 $F = 108.086$, 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 $r = .739, p < .001$, Spearman's $\rho = .748, p < .001$.

	Scores on data from human evaluation			
	Clarity	Readability	NIST	BLEU
SumTime	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.

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. 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 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 system is the clear winner, with no ratings of 1 and 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, for the 22 dates in the human evaluation only). In terms of means, SumTime 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 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 system and in the case of NIST, also above two of the PCFG systems (Table 3). The human and the automatic evaluations agree only that the PSCFG systems and PCFG-greedy are equally good.

7 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 SumTime domain of weather forecasts, to which we have added four 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 10 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 found that increasing the automatic component in system building from a handcrafted system to an automat-

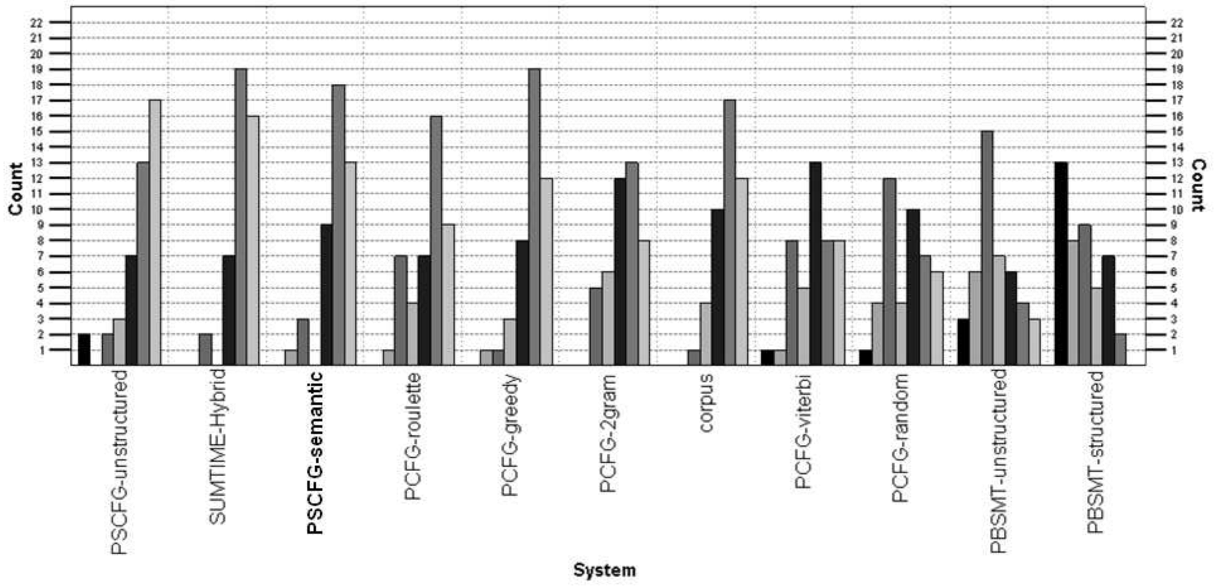

Figure 3: Clarity ratings: Number of times each system was rated 1, 2, 3, 4, 5, 6, and 7 on Clarity. Systems in descending order of mode (most frequent rating).

ically trained but manually crafted generator led to a loss of acceptability to human readers, but an improvement in terms of n-gram similarity to corpus texts. Further increasing the automatic component to the point where only a CFG for meaning representations is created manually did not result in a further reduction in quality in either acceptability to humans or corpus similarity. However, completely removing the manual component resulted in a reduction in quality in both human acceptability and corpus similarity (although this is more apparent in the former).

We found striking differences between the results from tests of human acceptability and measurements of corpus similarity. Compared to the human ratings, the automatic metrics severely underestimated the quality of the handcrafted SUMTIME 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 (Callison-Burch et al., 2006). From our results it seems clear that when the quality of diverse types of systems is compared, automatic metrics such as BLEU 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 the anonymous reviewers for their helpful comments.

References

A. Belz and E. Reiter. 2006. 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)*, pages 313–320.

A. Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

D. Chiang. 2006. An introduction to synchronous grammars (part of the course materials for the ACL'06 tutorial on synchronous grammars).

G. Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-

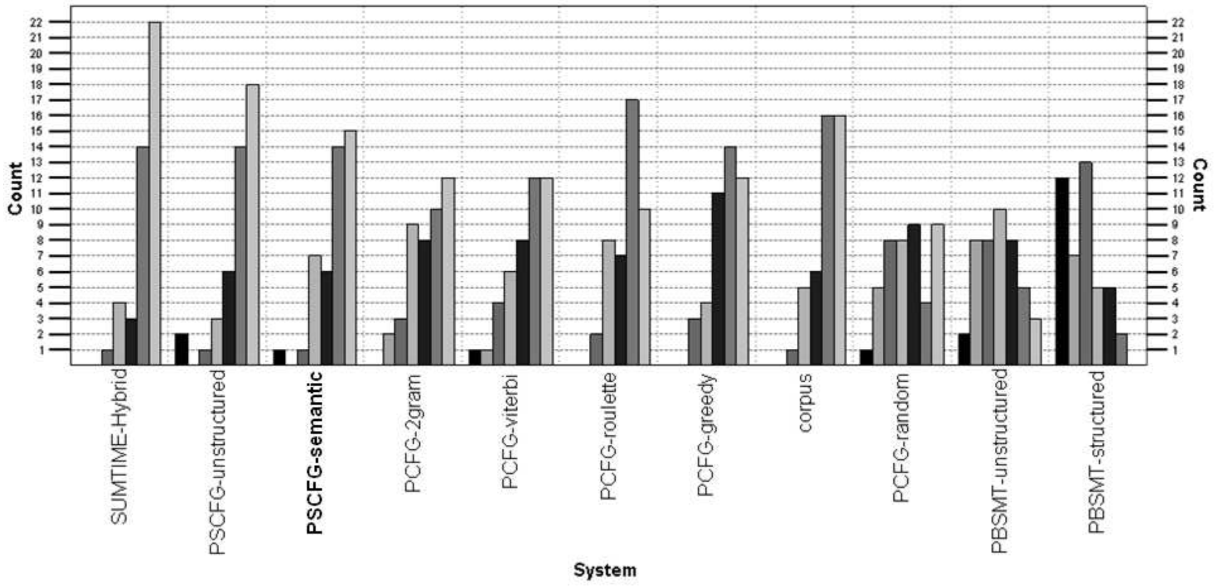

Figure 4: Readability ratings: Number of times each system was rated 1, 2, 3, 4, 5, 6, and 7 on Readability. Systems in descending order of mode (most frequent rating).

occurrence statistics. In *Proceedings of the ARPA Workshop on Human Language Technology*.

K. Knight and I. Langkilde. 1998. 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)*, pages 704–710.

P. Koehn, F. J. Och, and D. Marcu. 2003. 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)*, pages 48–54.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*, pages 177–180.

I. Langkilde. 2000. 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)*, pages 170–177.

F. J. Och and H. Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

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

E. Reiter and R. Dale. 1997. Building applied natural language generation systems. *Natural Langauge Engineering*, 3(1):57–87.

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

S. Riezler and J. T. Maxwell. 2005. 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*, pages 57–64.

S. Sripada, E. Reiter, J. Hunter, and J. Yu. 2002. SUMTIME-METEO: A parallel corpus of naturally occurring forecast texts and weather data. Technical Report AUCS/TR0201, Computing Science Department, University of Aberdeen.

Y. W. Wong and R. Mooney. 2006. 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)*, pages 439–446.

Y. W. Wong and R.J. Mooney. 2007. 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)*, pages 172–179.