Corpus-Driven Generation of Weather Forecasts

Anja Belz
Information Technology Research Institute
University of Brighton
Anja.Belz@itri.brighton.ac.uk

Abstract

In traditional natural language generation (NLG), careful analysis of a corpus of example texts and determining the single correct sublanguage behind it is seen as one of the main tasks of the NLG system builder. In practice, this often means elimination of variation in the corpus and specification of conditions for rule application to the point where an NLG system becomes (virtually) deterministic. This approach is time-consuming, does not apply objective criteria for deciding what is correct, and contributes to the lack of robustness and reusable components in NLG. Moreover, with variation regarded as a 'bug' to be eliminated, systems run the risk of implementing a subjective or restrictive view of the domain sublanguage. This research note argues that relative frequency provides an objective, easily applicable tool for dealing with corpus variation. The probabilistic approach to NLG can also help cut down on manual corpus analysis, make systems more robust and components more reusable. A methodology is described that combines use of a base generator with a separate, automatically adaptable, probabilistic decision-making component. Three different decision-making techniques are compared and evaluated with a focus on their ability to model idiolectal variation.

1 Introduction and Background

Examination of a corpus of example texts plays a fundamental role in building a language generation system for a given domain. Often, example texts are written by several authors using different terminology and style. Differences may concern content selection and discourse structure as well as choice of syntactic structure and lexical items. The traditional NLG system builder's task is typically seen as eliminating variation and writing generation rules with conditions for application precise enough to enable deterministic language generation.

McKeown *et al.* (1994) comment on “the tremendous variety in the possible sentences for each message type with respect to sentence structure and lexical choice” (p. 9) that the PLANDoc project team found in their domain corpus, and describe how one of “the two overriding practical considerations” was found to be “the need for a bounded sublanguage” (p. 8) which eliminated most of the variation found in the corpus.

Great emphasis is often placed on the importance of carrying out corpus analysis manually. Reiter and Dale (1997) recommend that NLG systems be built largely ”by careful analysis of the target text corpus, and by talking to domain experts” (p. 74, but also pp. 58, 61, 72 and 73), acknowledging that this may involve time and effort:

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

This is not only a time-consuming and expensive process (involving much experts' time), but also results in inherently domain-specific, non-portable and brittle systems. Moreover, idiolectal variation in the corpus is essentially regarded as noise: "Very speculatively, one could regard idiolect as a "bug" in the way language is used in the modern world."[1]

With notions of what is and is not correct in the sublanguage, and idiolectal variation regarded as noise or a bug, this view recalls Dummett's idea of the relation between idiolect and language, where:

> [...] the primary unit is still a *shared* language, known to all participants in a conversation; and the prototypical case is that in which they all use that language in the same way. (Dummett, 1994, p. 205).

In this view, the single shared language is acquired by individual speakers incompletely and imperfectly (Dummett, 1986), with each speaker's idiolect overlapping slightly different areas of the same shared, correct language.

The difficulty for the NLG system builder looking for the single shared and correct domain language is what criteria to apply in deciding which examples of usage in the corpus are correct (according to the rules of the domain language) and which are not. The recommended solution, 'ask the expert', is — apart from being expensive — unlikely to yield unanimous judgments (consistent and repeated variation being clear evidence that the expert authors of the corpus do not agree on rules of usage). Furthermore, there are NLG contexts where it is in fact more appropriate to emulate one particular idiolect, in order to simulate personality (e.g. for conversational agents), or simply because users have a preference for one of the idiolects.

This research note argues that it might be practically useful to take a more Fregean/Davidsonian view of idiolect and sublanguage: every, say, English speaker has their own individual language (their idiolect) which overlaps substantially with the languages of other English speakers (Frege, 1892; Davidson, 1986), but there is no single correct shared language. This implies regarding all the variation in the corpus as equally valid evidence of sublanguage usage, but focussing on the overlap between idiolects.

The principle of highest relative frequency provides a good objective handle on deciding appropriate usage: the problem of deciding what is correct is replaced with indentifying what is most likely. This enables the domain sublanguage to be modelled by estimating likelihood from the entire corpus, with probability mass naturally concentrated on usage in the overlap between idiolects. But it also enables individual idiolects to be modelled by estimating likelihood from a single-authored subsection of the corpus, without necessarily losing domain coverage, because the likelihood model can be smoothed for unseen usage from the rest of the corpus.

This approach applies an objective criterion for determining appropriate rules of usage, and offers a practical way of dealing with variation in the corpus. It also has the advantage of reducing the need for manual corpus analysis and consultation with experts, makes it easier to build portable system components (the same component can be trained on a range of different corpora), and provides robustness.

The approach to language generation outlined below, and previously described in Belz (2005), views all usage in the corpus as equally valid in principle. It combines use of a base genera-

[1] SumTime project summary, `http://www.csd.abdn.ac.uk/˜ereiter/sumtime.html`. 16/06/2005.

tor with a separate, automatically adaptable decision-making component. The starting point is to create a very general base generator capable of generating all the variation in a given corpus (not discriminating in terms of likelihood), and then to train it either on the entire corpus (modelling the sublanguage as a whole), or on a subcorpus corresponding to an idiolect with smoothing for unseen usage from the entire corpus. Three different decision-making techniques are compared and evaluated, in this report mainly in terms of their ability to model idiolect.

The following section describes the corpus of weather forecasts that was used in the experiments, and provides evidence of idiolectal variation in the corpus. Section 3 describes the base generator and the three techniques for training and running it that were compared in the experiments. Section 4 presents results for the three generators when trained on five different subcorpora as well as the entire corpus. Section 5 comments on the contribution of this research to NLG, and outlines directions for further research.

2 Domain and Data

2.1 Corpus of weather forecasts

The corpus used in the experiments reported below is the SumTime-Meteo corpus created by the SumTime project team in collaboration with WNI Oceanroutes (Sripada et al., 2002). The corpus was collected by WNI Oceanroutes from the commercial output of five different (human) forecasters, and each instance in the corpus consists of three numerical data files (generated by three different weather simulators) and the weather forecast file written by the forecaster on the evidence of the data files (and sometimes additional resources). Following the SumTime work, the experiments reported below focussed on the part of the forecasts that predicts wind characteristics for the next 15 hours. Such ‘wind statements’ look as follows:

```
2.FORECAST 1500 GMT FRI 10-Aug,TO 0600GMT SAT 11-Aug 2001
=====WARNINGS:              NIL                     =======
WIND(KTS)  CONFIDENCE: HIGH
  10M:     WNW-NW 12-15 BACKING W’LY 05-10 BY MIDNIGHT, THEN
           SW-SSW BY MORNING
  50M:     WNW-NW 15-18 BACKING W’LY 06-12 BY MIDNIGHT, THEN
           SW-SSW BY MORNING
```

To keep things simple, only the data file type that contains (virtually all) the information about wind parameters (the .tab file type) was used. The following is the part of the .tab file relevant to the above forecast.

```
O1, O2 AND O3 OIL FIELDS (EAST OF SHETLAND)
10-08-01

10/18  WNW  11  13  17  1.7  2.7  NW  1.5  7
10/21  W     8  10  12  1.5  2.4  NW  1.4  7
11/00  W     7   8  10  1.4  2.2  NW  1.4  7
11/03  SW    7   8  10  1.4  2.2  NW  1.3  7
11/06  SW    7   8  10  1.3  2.1  NW  1.3  7
...
```

The first column is the day/hour time stamp, the second the wind direction predicted for the corresponding time period; the third the wind speed at 10m above the ground; the fourth the gust speed at 10m; the fifth the gust speed at 50m. The remaining columns contain wave data.

The mapping from time series data to forecast is not straightforward (even when all three data files are taken into account). An example here is that while the wind direction in the first part of the wind statement is given as WNW-NW, NW does not appear as a wind speed anywhere in the data file. Nor is it obvious why the wind speeds 11, 8 and 7 are mapped to the two ranges 12-15 and 5-10.

For the experiments reported below, the SumTime-Meteo corpus was converted into a parallel corpus of wind statements and the wind data to be included in each statement. The wind data is a vector of time stamps and wind parameters, and was 'reverse-engineered', by automatically extracting information from both the forecast and the data file in parallel. The instances in the final training corpus look as follows (same example, 10-08-01, resulting in two input/output pairs):

```
WNW-NW 12 15 - - - W 05 10 - - 21 SW-SSW - - - - 03
WNW-NW 12-15 BACKING W'LY 05-10 BY MIDNIGHT THEN SW-SSW BY MORNING

SSW 05 10 - - - S - - - - - - 26 30 38 38 -1
SSW 05-10 BACKING S'LY AND INCREASING 26-30 GUSTS 38 BY END OF PERIOD
```

2.2 Weather forecasters' idiolects in the corpus

The converted corpus contains forecasts by five different meteorologists in almost identical numbers (about 410 each). The individual styles of the forecasters vary considerably. Reiter & Sripada (2002) have provided ample evidence of lexical and other variation in the SumTime-Meteo corpus including variation in time phrases. There also is interesting variation in 'transition' phrases, used to describe how wind direction and wind speed is changing, e.g. *S-SE 22-28* ***easing*** *20-25 by midday then* ***veering*** *and* ***decreasing*** *SW 8-12*. Of the near synonyms *easing*, *decreasing* and *falling*, all forecasters prefer the first to describe a drop in wind speed, although to different degrees. The picture is more mixed for near synonyms *rising*, *increasing* and *freshening*, with F1 and F5 strongly preferring the first, F2 and F3 exclusively using the second, and only F4 using *freshening*:

	F1	F2	F3	F4	F5
easing	128	316	215	246	149
decreasing	7	0	30	84	59
falling	15	0	19	0	0
rising	92	1	0	0	174
increasing	17	177	136	116	6
freshening	0	0	0	67	0

There are also significant differences that would concern earlier stages in an NLG system, such as text structuring and aggregation. As Table 1 shows, while F1 and F2 have a tendency to use conjunctions of speed and direction change verbs (e.g. ***veering and decreasing*** *SW 8-12*), F3, F4 and F5 hardly ever use such expressions[2]. The alternative, used more frequently by F3, F4

[2] Change verb conjunctions that are used at most once by any forecaster are not included in the table.

	F1	F2	F3	F4	F5	Total
veering and easing	9	18	0	1	0	28
easing and backing	0	18	0	2	4	24
backing and increasing	2	17	0	0	0	19
backing and easing	10	0	0	1	4	15
backing and falling	10	0	0	1	4	15
veering and falling	10	0	0	1	4	15
easing and veering	0	2	0	1	2	5
veering and increasing	0	5	0	0	0	5
backing and rising	4	0	0	0	0	4
decreasing and veering	0	0	0	0	2	2

Table 1: Frequency counts for transition phrases of pattern VERB-*ing and* VERB-*ing*.

and F5, is to keep wind direction and direction change verb, and wind speed and speed change verb together, e.g. *veering SW and decreasing 8-12*.

There are numerous examples of usage restricted to one or two idiolects, e.g. F3 never uses the change verb *becoming* (others: 23–48 times), only F4 uses *freshening* (67 times), only F4 uses constructions of change verbs separated by a slash, e.g. *backing/freshening*, more than twice (78 times). F3 and F5 use *S'ly* for *S* etc. at most once (others: 150–205 times).

The findings of these informally selected and analysed examples were complemented by more comprehensive numerical analysis. The following table gives the average forecast length, and the number of different 2-grams and 1-grams used by the five forecasters.

Forecaster	Avg length	2-grams	1-grams
F1	10.47	833	99
F2	8.23	739	96
F3	11.53	1019	107
F4	12.31	1074	116
F5	11.83	1065	115
All	10.85	2387	177

The average length of the forecasts ranges from 8.23 (F2) to 12.31 (F4), a substantial difference of more than four words. The total number of words used in the corpus is 177, but no forecaster uses more than two thirds of it. There is a difference of 19 words between the forecasters with the smallest and largest vocabularies. The differences in the numbers of 2-grams are correspondingly larger. Interestingly, there is an exact correlation between the differences in vocabulary size and those in word length: if a forecaster uses more words, then they will also produce longer forecasts.

To give another view of the degree of similarity between forecasters' styles, BLEU$_4$ scores were calculated for all pairings of forecasters and the corpus as a whole. The BLEU metric (Papineni et al., 2001) is ultimately based on n-gram agreement between sets of strings, with n a variable parameter. In simple terms, for BLEU with $n = 4$, the more 1-grams, 2-grams, 3-grams and 4-grams are the same between two strings, the higher their BLEU score. The sets of forecasts being compared here are not of the same size (unlike in the BLEU evaluation of the experiments below). The following table shows BLEU$_4$ similarity scores where one part of the corpus had the role of the gold standard set (indexing the rows) and the other that of the test set (indexing the columns)[3].

	F1	F2	F3	F4	F5	All
F1	1	**0.3042**	0.1332	0.2506	0.1768	0.1668
F2	**0.3077**	1	0.1248	0.2123	0.1480	0.1414
F3	0.1302	0.1163	1	**0.3599**	**0.3415**	0.2085
F4	0.2446	0.1972	**0.3599**	1	**0.3113**	0.2088
F5	0.1747	0.1409	**0.3420**	**0.3119**	1	0.1954
All	0.0096	0.0039	0.0275	0.0283	0.0212	1

From these similarity scores and the other comparisons above, some clear groups with high similarity scores (highlighted in the table in boldface font) emerge. F3, F4 and F5 use a similar size vocabulary, produce similar length forecasts and have a high BLEU$_4$ score (ranging from 0.3113 to 0.3599). F1 and F2 use a similar size vocabulary and use similar n-grams (BLEU$_4$ scores 0.3042 and 0.3077). There are secondary similarities between F1 and F4 (0.2506/0.2446), between F2 and F4 (0.2123/0.1972), but not between F1 or F2 and F3 or F5. In the comparisons of subcorpora F1-F5 to the corpus as a whole ('All'), F2 is the least similar to the whole corpus, and F3, F4 and F5 are the most similar. These results would predict e.g. that a generator trained on F1 would also do fairly well on F2 (and vice versa), and a generator trained on one of F3, F4 or F5 would also do well on the other two in this group.

3 NLG Methodology

The approach to language generation described in this section starts by creating a very general 'base' generator capable of generating all the variation in a given corpus, and then trains it on a given (sub)corpus, obtaining a probability distribution to descriminate between alternatives in terms of likelihood. Two different combinations of training technique and generation strategy have been compared and evaluated, one a standard 2-gram model as has been used in NLG before, the other two variations of a new type of probabilistic generator. This section starts by briefly reviewing existing work on statistical NLG, before describing the methodology used in the experiments below.

3.1 Statistical NLG

Traditional NLG systems tend to be domain-specific, not easily reusable or robust, as well as expensive to build. It was in order to address these issues that automatic methods for corpus exploitation, and probabilistic generation techniques based on them, started to be used in NLG about a decade ago.

Statistical NLG belongs to a group of generate-and-select approaches to NLG that are characterised by a separation between the definition of the space of all possible generation processes (the generation space) from the mechanism that controls which (set of) realisation(s) is selected as the output. Statistical generation from specified inputs started with Japan-Gloss (Knight et al., 1994; Knight et al., 1995), Nitrogen (Knight and Langkilde, 1998) and its successor Halogen (Langkilde, 2000).

[3] The BLEU score for two sets of strings of different sizes can vary depending on which plays the role of the gold standard (compare difference Recall/Precision).

In FERGUS, Bangalore *et al.* used an XTAG grammar to generate a word lattice representation of a small number of alternative realisations, and a 3-gram model to select the best (Bangalore and Rambow, 2000b). Humphreys *et al.* (2001) reused a PCFG trained for NL parsing to build syntactic generation trees from candidate syntactic nodes. Habash (2004) used structural 2-grams for lexical/syntactic selection tasks, as well as conventional n-grams for selection among surface strings. Velldal *et al.* (2004) compared a 4-gram model trained on the British National Corpus with a Maximum Entropy model reused from a parsing application in terms of their success in selecting correct realisations.

Some statistical NLG research has looked at subproblems of language generation, such as ordering of NP premodifiers (Shaw and Hatzivassiloglou, 1999; Malouf, 2000), attribute selection in content planning (Oh and Rudnicky, 2000), NP type determination (Poesio et al., 1999), pronominalisation (Strube and Wolters, 2000), and lexical choice (Bangalore and Rambow, 2000a).

Statistical NLG has mostly meant n-gram models, and has been applied either to all of surface realisation, or to a small subproblem in deep or surface realisation (but not to the entire generation process); it is either very expensive or not guaranteed to find the optimal solution; and the models it has used are either shallow and unstructured (like n-gram models), or require manual corpus annotation. The treebank-training approach described in Section 3.2.2 below can in principle be applied either to the entire generation process, or a subproblem, has been shown to be far less expensive than n-gram selection, and uses (structured) probability distributions over the entire generation space, not just the word sequences at its end.

3.2 Base generator, statistical training and generation techniques

A base generator for the converted weather forecasts corpus was written semi-automatically as a set of generation rules with atomic arguments that convert an input vector of numbers in steps to a set of NL forecasts. The automatic part was analysing the entire corpus with a set of simple chunking rules that split wind statements into wind direction, wind speed, gust speed, gust statements, time expressions, transition phrases (such as *and increasing*), pre-modifiers (such as *less than* for numbers, and *mainly* for wind direction), and post-modifiers (e.g. *in or near the low centre*). The manual part was to write the chunking rules themselves, and higher-level rules that combine different sequences of chunks into larger components.

Three different combinations of generation strategies and training techniques were tested: (i) a 2-gram model for selection among all realisations generated by the base generator, (ii) treebank-training the base generator and running it with a greedy generation strategy, and (iii) treebank-training the base generator and running it with a Viterbi generation strategy.

3.2.1 2-gram generator

In this set-up, the base generator was used directly to generate all alternative realisations for a given input, and the 2-gram model was applied to select the best realisation. The 2-gram model was a back-off model built using the SRILM toolkit, (Stolcke, 2002), and was used in more or less exactly the way reported in the Halogen papers (Langkilde, 2000). That is to say, the packed AND/OR-tree representation of all alternatives is generated in full, then the 2-gram model is applied to select the single best one. For more details see Belz (2005).

3.2.2 Treebank-training

In the other two set-ups, the base generators were treebank-trained on the (sub)corpus, then combined with a greedy and a Viterbi generation strategy. The remainder of this section first describes treebank-training of generators, then the two generation strategies.

The generation space of a generator can be seen as a set of decision trees, where the root nodes correspond to the inputs, and the paths down the trees are all possible generation processes (sequences of generator decisions) that lead to a realisation (leaf) node, a view discussed in Belz (2004).

The basic idea in generator treebank-training (described in more detail in Belz, 2005) is to estimate the likelihood of a realisation in terms of the likelihoods of the generator decisions that give rise to it, looking at the possible sequences of decisions that generate the realisation (its 'derivations'). One way of doing this is to say the likelihood of a string is the sum of the likelihoods of its derivations. To train such a model on a corpus of raw text, the set of derivations for each sentence is determined, frequencies for individual decisions are added up, and a probability distribution over sets of alternative decisions is estimated.

In the current version of the method, generation rules are context-free with atomic arguments. Derivations for sentences in the corpus are then standard context-free derivations, and corpora are annotated in the standard context-free way with brackets and labels. No disambiguation is performed, the assumption being that all derivations are equally good for a sentence. If a sentence has more than one derivation, frequency counts are divided equally between them. A standard maximum likelihood estimation is performed: the total number of occurrences of a decision (e.g. passive) is divided by the total number of occurrences of all alternatives (e.g. passive + active). The probability distribution is currently smoothed with the simple add-1 method. This is equivalent to Bayesian estimation with a uniform prior probability on all decisions, and is entirely sufficient for present purposes given the very small vocabulary and the good coverage of the data.

Greedy generation. The simplest strategy for using a treebank-trained generator during generation is to make the single most likely decision at each choice point in a generation process. This is not guaranteed to result in the most likely generation *process*, but the computational cost in application is exceedingly low.

Viterbi generation. The alternative is to do a Viterbi search of the generation forest for a given input, which maximises the joint likelihood of all decisions taken in the generation process. This is guaranteed to select the most likely generation process, but is considerably more expensive.

3.2.3 Preliminary comparison of generation methods

In related research reported elsewhere (Belz, 2005) the three generator types were compared and evaluated in detail. In sample experiments, the greedy generator was shown to take about one tenth of the time of the equivalent Viterbi generator, and the Viterbi generator to be still far more efficient than the equivalent 2-gram generator, taking about one seventh of the time. The 2-gram generator with its built-in bias towards shorter strings produced shorter realisations on average than the two treebank-trained generators. However, the 2-gram generator outperformed the other two on all tasks, although not by large margins. A random generator was used as a baseline which in the evaluation tests achieved a (cross-validated) BLEU_4 score of 0.215, compared to the greedy generator's 0.429, the Viterbi generator's 0.427, and the 2-gram generator's 0.453.

4 Experiments and Results

In the experiments, the 2-gram generator and the two treebank-trained generators were each trained on six different training sets (five subcorpora and the entire corpus), resulting in 18 generators to be tested altogether. These 18 generators were each tested on the training and testing parts of all six (sub)corpora, and results evaluated with the BLEU_4 evaluation metric, producing a total of 156 evaluation scores. The scores were five-fold cross-validated by repeating each test five times with a different random division into training and test set.

For any automatic learning method (counting statistical modelling among these for present purposes), there are two criteria under which results need to be assessed: specialisation and generalisation. On the one hand, a learning method needs to capture the training data well (specialisation), but without overfitting it, that is to say, it needs to generalise to a superset of similar data. In the present context, there are two types of similar data: one is the rest of the subcorpus data that the generator was not trained on (generalisation from the training examples of the idiolect to the entire idiolect), and the other is the data in the rest of the entire corpus (generalisation from idiolect to domain sublanguage).

In the remainder of this section, results will be assessed from three points of view: (i) specialisation and generalisation to the idiolect — how well do the subcorpus-trained generators do on their own training and test sets, and how much better than the generators trained on the entire corpus; (ii) domain generalisation — how do the subcorpus-trained generators do on the entire corpus, compared to each other as well as to the generators trained on the entire corpus; and (iii) similarity across subcorpora — how do the subcorpus-trained generators perform on the other subcorpora.

4.1 Specialisation and test set generalisation

The first table below shows results for each of the six training sets. The columns are indexed by the training sets, the rows by generators: greedy/All, viterbi/All and 2-gram/All are the three generators trained on the entire corpus; greedy/same, viterbi/same and 2-gram/same refers to the generators trained on the same training set as shown in the column header. For example, the greedy/same row shows results for the greedy generator when trained on the entire corpus in the All-train column, for the greedy generator trained on the F1 subcorpus in the F1-train column, for the greedy generator trained on the F2 subcorpus in the F2-train column, and so on.

If idiolect specialisation is good, then a subcorpus-trained generator should do better than the equivalent generator trained on the entire corpus when applied to the test set of their own subcorpus. Training set specialisation can be seen as the minimum requirement for an idiolect modelling technique, and results show that the subcorpus-trained generators do better than their counterparts trained on the entire corpus in every single case, although sometimes by smaller, sometimes by larger margins. For each corpus (column), the best result achieved by any generator is highlighted in bold, here without exception achieved by the 2-gram generator.

	All-train	F1-train	F2-train	F3-train	F4-train	F5-train
greedy/All	0.4377	0.4126	0.5526	0.4066	0.4136	0.4182
greedy/same	0.4377	0.4288	0.5714	0.4776	0.4205	0.4816
viterbi/All	0.4346	0.4119	0.5620	0.4045	0.4112	0.4121
viterbi/same	0.4346	0.4286	0.5796	0.4778	0.4193	0.4774
2-gram/All	**0.4603**	0.4330	0.5681	0.4351	0.4253	0.4616
2-gram/same	**0.4603**	**0.4662**	**0.6107**	**0.5405**	**0.4595**	**0.5055**

The next table below shows results for the six test sets, assessing how well each of the generators performed on the test set, i.e. the part of the subcorpus that it was not trained on (idiolect generalisation). Here, the columns are indexed by the test sets, and the rows are indexed as before, e.g. greedy/same for F1-test means the test set results for the greedy generator trained on the training part of the F1 subcorpus. If generalisation from the training set to the idiolect is good then the number in the */same row should be higher than the number in the */All row immediately above it. These figures can be seen as a test that the idiolect models have succeeded in their main task, namely to model the entire idiolect.

With one exception (2-gram model on F1), the subcorpus-trained models all do better on their own test sets than the models trained on the entire corpus. The average percentage improvement is better for the treebank-trained generators, indicating that the 2-gram generators specialise more on the training set. As expected, the test set results are, on the whole, slightly worse than the training set results above.

	All-test	F1-test	F2-test	F3-test	F4-test	F5-test
greedy/All	0.4299	0.3978	0.5567	0.3992	0.4107	0.4060
greedy/same	0.4299	0.4290	0.5716	0.4642	0.4121	**0.4743**
viterbi/All	0.4274	0.3992	0.5624	0.3965	0.4094	0.4014
viterbi/same	0.4274	0.4326	0.5779	0.4629	0.4151	0.4712
2-gram/All	**0.4536**	**0.4363**	0.5631	0.4164	0.4284	0.4545
2-gram/same	**0.4536**	0.4312	**0.5980**	**0.5050**	**0.4416**	0.4712

Best results for a corpus are again highlighted. The same-corpus-trained 2-gram generator outperforms the others on All-test, F2-test, F3-test and F4-test, the all-corpus-trained 2-gram generator performs best on F1-test, and the subcorpus-trained Viterbi generator on F5-test. This is in contrast to the training set results above, where the subcorpus-trained 2-gram generator outperforms the others by fairly substantial margins in all cases.

4.2 Generalisation to the entire corpus

The next set of results assesses domain generalisation by comparing the performance of each of the subcorpus-trained generators on the entire corpus (for ease of comparison scores for the corresponding generators trained on the entire corpus are repeated in the first column). If domain generalisation is good, then the scores for a subcorpus-trained generator should be close to those of the corresponding */All generator. The results for the two treebank-trained generators show that they achieve domain generalisation extremely well, but the surprising result is that except for greedy/F2 and viterbi/F2, all the subcorpus-trained greedy and Viterbi generators actually do as well as, or (in most cases) *better* than greedy/All and viterbi/All, in the case of greedy/F3 and viterbi/F3 by substantial margins.

	greedy/All	greedy/F1	greedy/F2	greedy/F3	greedy/F4	greedy/F5
All-test	0.4299	0.4297	0.4165	**0.4541**	0.4312	0.4394
	viterbi/All	viterbi/F1	viterbi/F2	viterbi/F3	viterbi/F4	viterbi/F5
All-test	0.4274	0.4279	0.4153	***0.4660***	0.4294	0.4378
	2-gram/All	2-gram/F1	2-gram/F2	2-gram/F3	2-gram/F4	2-gram/F5
All-test	0.4536	0.3980	0.4160	**0.4605**	0.4380	0.4200

In contrast, all the subcorpus-trained 2-gram generators except 2-gram/F3 perform significantly worse than 2-gram/All. Combined with the fact that idiolect generalisation was also slightly worse for the 2-gram generators, this indicates that the latter overfit the training data to some extent.

It is in all three cases the F3-trained generator that performs best on All-test, with viterbi-F3 the overall winner by a small margin. It is difficult to evaluate an isolated result of this kind. However, the fact that two such very different training techniques as 2-gram modelling and treebank-training both do best when trained on subcorpus F3 might be taken to indicate that F3 is highly representative of the corpus as a whole (in the BLEU comparison above, F3 had the second highest similarity to the entire corpus).

4.3 Similarity across subcorpora

The final set of results below looks at how well the subcorpus-trained generators perform on the test sets of the subcorpora that they were not trained on (again, for ease of comparison some results are repeated from above). Here, what would be expected is for each generator to achieve a better score on its own subcorpus than the other generators. Furthermore, generators trained on more similar subcorpora should do better on each other's subcorpus. As for the first expectation, it is confirmed in most cases but not all: among the greedy generators, greedy/F5 (not greedy/F1) performs best on F1-test; among the Viterbi generators, viterbi/F5 (not viterbi/F1) is best on F1-test; and among the 2-gram generators, 2-gram/All (not 2-gram/F1) is best on F1-test. Greedy/F3 and viterbi/F3 are better on F4-test than greedy/F4 and viterbi/F4, respectively.

	greedy/All	greedy/F1	greedy/F2	greedy/F3	greedy/F4	greedy/F5
F1-test	0.3978	0.4290	0.3985	0.4259	0.3953	**0.4337**
F2-test	0.5567	0.5437	**0.5716**	0.5096	0.5405	0.5402
F3-test	0.3992	0.3701	0.3745	**0.4642**	0.4095	0.3635
F4-test	0.4107	0.4072	0.3885	**0.4351**	0.4121	0.4023
F5-test	0.4060	0.4268	0.3838	0.4372	0.4076	***0.4743***
	viterbi/All	viterbi/F1	viterbi/F2	viterbi/F3	viterbi/F4	viterbi/F5
F1-test	0.3992	0.4326	0.4024	0.4337	0.3967	***0.4384***
F2-test	0.5624	0.5464	**0.5779**	0.5381	0.5527	0.5426
F3-test	0.3965	0.3704	0.3732	**0.4629**	0.4057	0.3639
F4-test	0.4094	0.4006	0.3852	***0.4481***	0.4151	0.4001
F5-test	0.4014	0.4223	0.3809	0.4440	0.4031	**0.4712**
	2-gram/All	2-gram/F1	2-gram/F2	2-gram/F3	2-gram/F4	2-gram/F5
F1-test	**0.4363**	0.4312	0.4033	0.4087	0.4205	0.3985
F2-test	0.5631	0.5287	***0.5980***	0.4966	0.542	0.4866
F3-test	0.4164	0.3401	0.3563	***0.5050***	0.3889	0.3751
F4-test	0.4284	0.3564	0.3808	0.4280	**0.4416**	0.3703
F5-test	0.4545	0.3746	0.3898	0.4474	0.4242	**0.4712**

In absolute terms, the best score on the F1 test set was achieved by the viterbi/F5 generator, on the F2 test set by the 2-gram/F2 generator, on F3-test by 2-gram/F3, on F4-test by viterbi/F3, on F5-test by greedy/F5 and on the entire corpus by viterbi/F3 (all highlighted in the above two tables in bold italics). The three generator types perform virtually identically when average scores are calculated on the above three tables — greedy: 0.437, mean deviation: 0.046; Viterbi: 0.439, mean deviation: 0.050; 2-gram: 0.435, mean deviation: 0.049. Performance is simply distributed differently.

The comparisons in Section 2.2 above predicted that generators trained on F3, F4 and F5 would do well on each other's subcorpus, as would generators trained on F1, F2 and F4, and that generators trained on F1 and F5, F1 and F3, F2 and F3, and F2 and F5 would do badly on each other's subcorpora. These predictions are very clearly confirmed in the 2-gram results, which is not altogether surprising, since the BLEU metric is ultimately based on n-gram agreement.

As for the treebank-trained generators, the only result clearly predicted by the comparisons are that greedy/F3 and greedy/F4 would perform well on each other's corpora. This confirms that the generator treebank-training method exploits some very different regularities and frequencies in the corpus than n-gram modelling. While it too exploits the frequencies of word sequences and word sequence cooccurrences in raw text, it distributes these over the entire generation space of generator decisions, instead of deriving a model of surface string likelihood. The likelihoods in the treebank-trained generators are attached to sets of strings instead of (sub)strings, so generators trained on subcorpora with low n-gram similarity scores such as F1 and F5 might still do well on each other's test sets, as long as many of their substrings are in the same sets. It may be the case that the similarities exploited by generator treebank-training and n-gram models are largely complementary, in which case a method combining the two might outperform them (a topic for future research).

5 Conclusions and Further Research

The research presented here is part of an ongoing research project[4] the purpose of which is to investigate issues in generic NLG. The focus of this paper is on development and comparison of statistical methodology for NLG, in particular with respect to idiolect modelling. It was not the primary objective to optimise the weather forecasts for a real-life application, e.g. very little effort was put in writing the base generator.

While the three methods achieved similar scores on average, treebank-training was shown to be a more 'distributed' form of learning, not overfitting the data as much as n-gram modelling, and achieving better generalisaton from subcorpus to corpus. Combined with the fact that the treebank-trained generators are also far less expensive to apply, and do not have the n-gram model's almost absolute built-in bias towards longer realisations, these results make generator treebank-training look like a promising way forward for probabilistic language generation.

The research reported in this paper should not be taken as intending to replace manual corpus analysis. In order to write a base generator as described above, domain knowledge is still needed, the corpus still needs to be examined. However, the main effort in identifying sublanguages for non-probabilistic NLG systems lies in creating a sublanguage specification that

[4] Controlled Generation of Text (CoGenT) `http://www.itri.brighton.ac.uk/projects/cogent`.

permits (near) deterministic generation. This is a tall order because it means working out generation rules that generate only one (or close to one) realisation for every given combination of system input and generation context. In practice, as McKeown *et al.* (1994) comment, this often means eliminating variation to the point where the task becomes manageable. It is the determination of conditions for rule application, and the problem of deciding which usage is correct and which is incorrect that probabilistic NLG in general proposes to replace. Generator treebank-training is a promising alternative to n-gram modelling which has so far dominated probabilistic NLG.

Acknowledgements

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

References

S. Bangalore and O. Rambow. 2000a. Corpus-based lexical choice in natural language generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL '00)*, pp 464–471.

S. Bangalore and O. Rambow. 2000b. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING '00)*, pp 42–48.

A. Belz. 2004. Underspecification for NLG. In *INLG04 Posters: Extended Abstracts of Posters Presented at the Third International Conference on Natural Language Generation*. Technical Report ITRI-04-01, ITRI, University of Brighton.

A. Belz. 2005. Statistical generation: Three methods compared and evaluated. In *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG'05)*. To appear.

D. Davidson. 1986. A nice derangement of epitaphs. In E. LePore (ed), *Truth and Interpretation*, pp 433–446. Blackwell, Oxford.

M. Dummett. 1986. A nice derangement of epitaphs: Some comments on Davidson and Hacking. In E. LePore (ed), *Truth and Interpretation*, pp 459–476. Blackwell, Oxford.

M. Dummett. 1994. Reply to Davidson. In B. McGuinness and G. Oliveri (eds), *The Philosophy of Michael Dummett*, pp 257–267. Kluwer.

G. Frege. 1892. Über Sinn und Bedeutung. In M. Beaney (ed), *The Frege Reader*. Blackwell, Oxford. This publication 1997.

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

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

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 the 17th International Conference on Computational Linguistics (COLING-ACL '98)*, pp 704–710.

K. Knight, I. Chander, M. Haines, V. Hatzivassiloglou, E. Hovy, M. Iida, S. Luk, A. Okumura, R. Whitney and K. Yamada. 1994. Integrating knowledge bases and statistics in MT. In *Proceedings of the 1st Conference of the Association for Machine Translation in the Americas (AMTA '94)*, pp 134–141.

K. Knight, I. Chander, M. Haines, V. Hatzivassiloglou, E. Hovy, M. Iida, S. Luk, R. Whitney and K. Yamada. 1995. Filling knowledge gaps in a broad-coverage MT system. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI '95)*, pp 1390–1397.

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)*, pp 170–177.

R. Malouf. 2000. The order of prenominal adjectives in natural language generation. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL '00)*, pp 85–92.

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

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

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.

M. Poesio, R. Henschel, J. Hitzeman and R. Kibble. 1999. Statistical NP generation: A first report. In R. Kibble and K. van Deemter (eds), *Proceedings of ESSLLI Workshop on NP Generation*, pp 30–42.

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

E. Reiter and S. Sripada. 2002. Human variation and lexical choice. *Computational Linguistics*, 28:545–553.

J. Shaw and V. Hatzivassiloglou. 1999. Ordering among premodifiers. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL '99)*, pp 135–143.

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

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

M. Strube and M. Wolters. 2000. A probabilistic genre-independent model of pronominalization. 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 18–25.

E. Velldal, S. Oepen and D. Flickinger. 2004. Paraphrasing treebanks for stochastic realization ranking. In *Proceedings of the 3rd Workshop on Treebanks and Linguistic Theories (TLT '04)*, Tuebingen, Germany.