Exploiting Multi-Word Units in History-Based Probabilistic Generation

Deirdre Hogan, Conor Cafferkey, Aoife Cahill* and **Josef van Genabith**
National Centre for Language Technology
School of Computing, Dublin City University
Dublin 9, Ireland
dhogan,ccafferkey,josef@computing.dcu.ie

Abstract

We present a simple history-based model for sentence generation from LFG f-structures, which improves on the accuracy of previous models by breaking down PCFG independence assumptions so that more f-structure conditioning context is used in the prediction of grammar rule expansions. In addition, we present work on experiments with named entities and other multi-word units, showing a statistically significant improvement of generation accuracy. Tested on section 23 of the Penn Wall Street Journal Treebank, the techniques described in this paper improve BLEU scores from 66.52 to 68.82, and coverage from 98.18% to 99.96%.

1 Introduction

Sentence generation, or surface realisation, is the task of generating meaningful, grammatically correct and fluent text from some abstract semantic or syntactic representation of the sentence. It is an important and growing field of natural language processing with applications in areas such as transfer-based machine translation (Riezler and Maxwell, 2006) and sentence condensation (Riezler et al., 2003). While recent work on generation in restricted domains, such as (Belz, 2007), has shown promising results there remains much room for improvement particularly for broad coverage and robust generators, like those of Nakanishi et al. (2005) and Cahill and van Genabith (2006), which do not rely on handcrafted grammars and thus can easily be ported to new languages.

This paper is concerned with sentence generation from Lexical-Functional Grammar (LFG) f-structures (Kaplan, 1995). We present improvements in previous LFG-based generation models firstly by breaking down PCFG independence assumptions so that more f-structure conditioning context is included when predicting grammar rule expansions. This history-based approach has worked well in parsing (Collins, 1999; Charniak, 2000) and we show that it also improves PCFG-based generation.

We also present work on utilising named entities and other multi-word units to improve generation results for both accuracy and coverage. There has been a limited amount of exploration into the use of multi-word units in probabilistic parsing, for example in (Kaplan and King, 2003) (LFG parsing) and (Nivre and Nilsson, 2004) (dependency parsing). We are not aware of any similar work on generation. In the LFG-based generation algorithm presented by Cahill and van Genabith (2006) complex named entities (i.e. those consisting of more than one word token) and other multi-word units can be fragmented in the surface realization. We show that the identification of such units may be used as a simple measure to constrain the generation model's output.

We take the generator of (Cahill and van Genabith, 2006) as our baseline generator. When tested on f-structures for all sentences from Section 23 of the Penn Wall Street Journal (WSJ) treebank (Mar-

* Now at the Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart, Azenbergstrae 12, D-70174 Stuttgart, Germany. aoife.cahill@ims.uni-stuttgart.de

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 267–276, Prague, June 2007.

cus et al., 1993), the techniques described in this paper improve BLEU score from 66.52 to 68.82. In addition, coverage is increased from 98.18% to almost 100% (99.96%).

The remainder of the paper is structured as follows: in Section 2 we review related work on statistical sentence generation. Section 3 describes the baseline generation model and in Section 4 we show how the new history-based model improves over the baseline. In Section 5 we describe the source of the multi-word units (MWU) used in our experiments and the various techniques we employ to make use of these MWUs in the generation process. Section 6 gives experimental details and results.

2 Related Work on Statistical Generation

In (statistical) generators, sentences are generated from an abstract linguistic encoding via the application of grammar rules. These rules can be hand-crafted grammar rules, such as those of (Langkilde-Geary, 2002; Carroll and Oepen, 2005), created semi-automatically (Belz, 2007) or, alternatively, extracted fully automatically from treebanks (Bangalore and Rambow, 2000; Nakanishi et al., 2005; Cahill and van Genabith, 2006).

Insofar as it is a broad coverage generator, which has been trained and tested on sections of the WSJ corpus, our generator is closer to the generators of (Bangalore and Rambow, 2000; Langkilde-Geary, 2002; Nakanishi et al., 2005) than to those designed for more restricted domains such as weather forecast (Belz, 2007) and air travel domains (Ratnaparkhi, 2000).

Another feature which characterises statistical generators is the probability model used to select the most probable sentence from among the space of all possible sentences licensed by the grammar. One generation technique is to first generate all possible sentences, storing them in a word lattice (Langkilde and Knight, 1998) or, alternatively, a generation forest, a packed represention of alternate trees proposed by the generator (Langkilde, 2000), and then select the most probable sequence of words via an n-gram language model.

Increasingly syntax-based information is being incorporated directly into the generation model. For example, Carroll and Oepen (2005) describe a sentence realisation process which uses a hand-crafted HPSG grammar to generate a generation forest. A selective unpacking algorithm allows the extraction of an *n*-best list of realisations where realisation ranking is based on a maximum entropy model. This unpacking algorithm is used in (Velldal and Oepen, 2005) to rank realisations with features defined over HPSG derivation trees. They achieved the best results when combining the tree-based model with an *n*-gram language model.

Nakanishi et al. (2005) describe a treebank-extracted HPSG-based chart generator. Importing techniques developed for HPSG parsing, they apply a log linear model to a packed representation of all alternative derivation trees for a given input. They found that a model which included syntactic information outperformed a bigram model as well as a combination of bigram and syntax model.

The probability model described in this paper also incorporates syntactic information, however, unlike the discriminative HPSG models just described, it is a generative history- and PCFG-based model. While Belz (2007) and Humphreys et al. (2001) mention the use of contextual features for the rules in their generation models, they do not provide details nor do they provide a formal probability model. To the best of our knowledge this is the first paper providing a probabilistic generative, history-based generation model.

3 Surface Realisation from f-Structures

Cahill and van Genabith (2006) present a probabilistic surface generation model for LFG (Kaplan, 1995). LFG is a constraint-based theory of grammar, which analyses strings in terms of c(onstituency)-structure and f(unctional)-structure (Figure 1). C-structure is defined in terms of CFGs, and f-structures are recursive attribute-value matrices which represent abstract syntactic functions (such as SUBJect, OBJect, OBLique, COMPlement (sentential), ADJ(N)unct), agreement, control, long-distance dependencies and some semantic information (e.g. tense, aspect).

C-structures and f-structures are related in a projection architecture in terms of a piecewise correspondence ϕ.[1] The correspondence is indicated in

[1] Our formalisation follows (Kaplan, 1995).

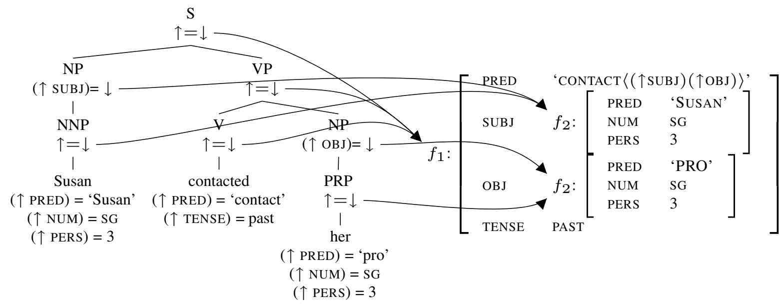

Figure 1: C- and f-structures with ϕ links for the sentence *Susan contacted her.*

terms of the curvy arrows pointing from c-structure nodes to f-structure components in Figure 1. Given a c-structure node n_i, the corresponding f-structure component f_j is $\phi(n_i)$. F-structures and the c-structure/f-structure correspondence are described in terms of functional annotations on c-structure nodes (CFG grammar rules). An equation of the form (↑F) = ↓ states that the f-structure associated with the mother of the current c-structure node (↑) has an attribute (grammatical function) (F), whose value is the f-structure of the current node (↓). The up-arrows and down-arrows are shorthand for $\phi(\mathrm{M}(n_i)) = \phi(n_i)$ where n_i is the c-structure node annotated with the equation.[2]

$$Tree_{best} := \text{argmax}_{Tree} P(Tree|F\text{-}Str) \tag{1}$$

$$P(Tree|F\text{-}Str) := \prod_{\substack{X \rightarrow Y \; in \; Tree \\ Feats = \{a_i | \exists v_j (\phi(X)) a_i = v_j\}}} P(X \rightarrow Y | X, Feats) \tag{2}$$

The generation model of (Cahill and van Genabith, 2006) maximises the probability of a tree given an f-structure (Eqn. 1), and the string generated is the yield of the highest probability tree. The generation process is guided by *purely* local information in the input f-structure: f-structure annotated CFG rules (LHS → RHS) are conditioned on their LHSs and on the set of features/attributes *Feats* = $\{a_i | \exists v_j \phi(\text{LHS}) a_i = v_j\}$[3] ϕ-linked to the LHS (Eqn. 2). Table 1 shows a generation grammar rule and conditioning features extracted from the example in Figure 1. The probability of a tree is decomposed into the product of the probabilities of the f-structure annotated rules (conditioned on the LHS and local *Feats*) contributing to the tree. Conditional probabilities are estimated using maximum likelihood estimation.

grammar rule	local conditioning features
S(↑=↓)→ NP(↑SUBJ=↓) VP(↑=↓)	S(↑=↓), {SUBJ,OBJ,PRED,TENSE}

Table 1: Example grammar rule (from Figure 1).

Cahill and van Genabith (2006) note that conditioning f-structure annotated generation rules on local features (Eqn. 2) can sometimes cause the model to make inappropriate choices. Consider the following scenario where in addition to the c-/f-structure in Figure 1, the training set contains the c-/f-structure displayed in Figure 2.

From Figures 1 and 2, the model learns (among others) the generation rules and conditional probabilities displayed in Tables 2 and 3.

F-Struct Feats	Grammar Rules	Prob
{SUBJ, OBJ, PRED}	S(↑=↓) → NP(↑SUBJ=↓) VP(↑=↓)	1
{SUBJ, OBJ, PRED}	VP(↑=↓) → V(↑=↓) NP(↑OBJ=↓)	1
{NUM, PER, GEN}	NP(↑SUBJ=↓) → NNP(↑=↓)	0.5
{NUM, PER, GEN}	NP(↑SUBJ=↓) → PRP(↑=↓)	0.5
{NUM, PER, GEN}	NP(↑OBJ=↓) → PRP(↑=↓)	1

Table 2: A sample of internal grammar rules extracted from Figures 1 and 2.

Given the input f-structure (for `She accepted`) in Figure 3, (and assuming suitable generation rules for intransitive VPs and `accepted`) the model would produce the inappropriate highest probability tree of Figure 4 with an incorrect case for the pronoun in subject position.

[2] M is the mother function on CFG tree nodes.

[3] In words, *Feats* is the set of top level features/attributes (those attributes a_i for which there is a value v_i) of the f-structure ϕ linked to the LHS.

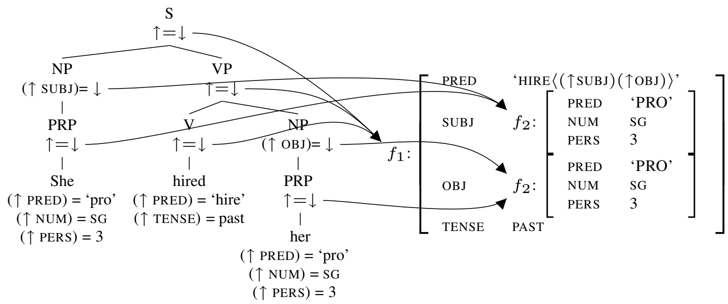

Figure 2: C- and f-structures with ϕ links for the sentence *She hired her.*

F-Struct Feats	Grammar Rules	Prob
{PRED=PRO,NUM=SG PER=3, GEN=FEM}	PRP(↑=↓) → she	0.33
{PRED=PRO,NUM=SG PER=3, GEN=FEM}	PRP(↑=↓) → her	0.66

Table 3: A sample of lexical item rules extracted from Figures 1 and 2.

$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{PRED} & \text{pro} \\ \text{NUM} & \text{sg} \\ \text{PERS} & 3 \\ \text{GEND} & \text{fem} \end{bmatrix} \\ \text{PRED} & \text{accept} \\ \text{TENSE} & \text{past} \end{bmatrix}$$

Figure 3: Input f-structure for `She accepted`.

To solve the problem, Cahill and van Genabith (2006) apply an automatic generation grammar transformation to their training data: they automatically label CFG nodes with additional case information and the model now learns the new improved generation rules of Tables 4 and 5. Note how the additional case labelling subverts the problematic independence assumptions of the probability model and communicates the fact that a subject NP has to be realised as nominative case from the *S → NP-nom VP* production, via the intermediate *NP-nom → PRP-nom*, down to the lexical production *PRP-nom → she*. The labelling guarantees that, given the example f-structure in Figure 3, the model generates the correct string `she accepted`.

F-Struct Feats	Grammar Rules
{SUBJ, OBJ, PRED}	S(↑=↓) → NP-nom(↑SUBJ=↓) VP(↑=↓)
{SUBJ, OBJ, PRED}	VP(↑=↓) → V(↑=↓) NP-acc(↑OBJ=↓)
{NUM, PER, GEN}	NP-nom(↑SUBJ=↓) → PRP-nom(↑=↓)
{NUM, PER, GEN}	NP-nom(↑SUBJ=↓) → NNP-nom(↑=↓)
{NUM, PER, GEN}	NP-acc(↑OBJ=↓) → PRP-acc(↑=↓)

Table 4: Internal grammar rules with case markings.

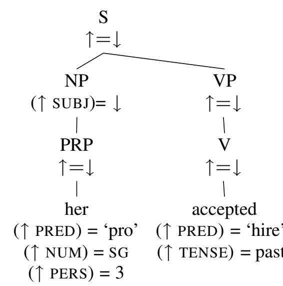

Figure 4: Inappropriate output: *her accepted.*

F-Struct Feats	Grammar Rules
{PRED=PRO,NUM=SG PER=3, GEN=FEM}	PRP-nom(↑=↓) → she
{PRED=PRO,NUM=SG PER=3, GEN=FEM}	PRP-acc(↑=↓) → her

Table 5: Lexical item rules with case markings

4 A History-Based Generation Model

The automatic generation grammar transform presented in (Cahill and van Genabith, 2006) provides a solution to coarse-grained and (in fact) inappropriate independence assumptions in the basic generation model. However, there is a sense in which the proposed cure improves on the symptoms, but not the cause of the problem: it weakens independence assumptions by multiplying and hence increasing the specificity of conditioning CFG category labels. There is another option available to us, and that is the option we will explore in this paper: instead of applying a generation grammar transform, we will improve the f-structure-based conditioning of the generation rule probabilities. In the original model, rules are conditioned on *purely local* f-structure context: the set of features/attributes ϕ-linked to the LHS of a grammar rule. As a direct consequence of this, the conditioning (and hence the model) cannot not distinguish between NP, PRP and NNP rules appropriate to e.g. subject (SUBJ) or object contexts (OBJ) in a given input f-structure. However, the required information can easily be incorporated into the generation model by uniformly conditioning generation rules on their *parent* (*mother*) grammatical function, in addition to the local ϕ-linked feature set. This additional conditioning has the effect of making the choice of generation rules sensitive to the *history* of the generation process, and, we argue, provides a simpler, more uniform, general, intuitive and natural probabilistic generation model obviating the need for CFG-grammar transforms in the original proposal of (Cahill and van Genabith, 2006).

In the new model, each generation rule is now conditioned on the LHS rule CFG category, the set of features ϕ-linked to LHS *and* the parent grammatical function of the f-structure ϕ-linked to LHS. In a given c-/f-structure pair, for a CFG node n, the parent grammatical function of the f-structure ϕ-linked to n is that grammatical function GF, which, if we take the f-structure ϕ-linked to the mother M(n), and apply it to GF, returns the f-structure ϕ-linked to n: $(\phi(\mathrm{M}(n))\mathrm{GF}) = \phi(n)$.

The basic idea is best explained by way of an example. Consider again Figure 1. The mother grammatical function of the f-structure f_2 associated with node NP(↑SUBJ=↓) and its daughter NNP(↑=↓) (via the ↑=↓ functional annotation) is SUBJ, as $(\phi(\mathrm{M}(n_2))\mathrm{SUBJ}) = \phi(n_2)$, or equivalently $(f_1\mathrm{SUBJ}) = f_2$.

Given Figures 1 and 2 as training set, the improved model learns the generation rules (the mother grammatical function of the outermost f-structure is assumed to be a dummy TOP grammatical function) of Tables 6 and 7.

F-Struct Feats	Grammar Rules
{SUBJ, OBJ, PRED, **TOP**}	S(↑=↓) → NP(↑SUBJ=↓) VP(↑=↓)
{SUBJ, OBJ, PRED, **TOP**}	VP(↑=↓) → V(↑=↓) NP(↑OBJ=↓)
{NUM, PER, GEN, **SUBJ**}	NP(↑SUBJ=↓) → PRP(↑=↓)
{NUM, PER, GEN, **OBJ**}	NP(↑OBJ=↓) → PRP(↑=↓)
{NUM, PER, GEN, **SUBJ**}	NP(↑SUBJ=↓) → NNP(↑=↓)

Table 6: Grammar rules with extra feature extracted from F-Structures.

Note, that for our example the effect of the uniform additional conditioning on mother grammatical function has the same effect as the generation grammar transform of (Cahill and van Genabith, 2006), but without the need for the grammar transform. Given the input f-structure in Figure 3, the model will generate the correct string `she accepted`. In addition, uniform conditioning on mother grammatical function is more general than the case-phenomena specific generation grammar transform of (Cahill and van Genabith, 2006), in that it applies to each and every sub-part of a recursive input f-structure driving generation, making available relevant generation history (context) to guide local generation decisions.

F-Struct Feats	Grammar Rules
{PRED=PRO,NUM=SG PER=3, GEN=FEM, **SUBJ**}	PRP(↑=↓) → she
{PRED=PRO,NUM=SG PER=3, GEN=FEM, **OBJ**}	PRP(↑=↓) → her

Table 7: Lexical item rules.

The new history-based probabilistic generation model is defined as:

$$P(Tree|\textit{F-Str}) := \prod_{\substack{X \rightarrow Y \; in \; Tree \\ Feats = \{a_i | \exists v_j (\phi(X)) a_i = v_j\} \\ (\phi(\mathrm{M}(X)))\mathbf{GF} = \phi(X)}} P(X \rightarrow Y | X, Feats, \mathbf{GF}) \quad (3)$$

Note that the new conditioning feature, the f-structure mother grammatical function, GF, is available from structure previously generated in the c-structure tree. As such, it is part of the *history* of the tree, i.e. it has already been generated in the top-down derivation of the tree. In this way, the generation model resembles history-based models for parsing (Black et al., 1992; Collins, 1999; Charniak, 2000). Unlike, say, the parent annotation for parsing of (Johnson, 1998) the parent GF feature for a particular node expansion is not merely extracted from the parent node in the c-structure tree, but is sometimes extracted from an ancestor node further up the c-structure tree via intervening ↑=↓ functional annotations.

Section 6 provides evaluation results for the new model on section 23 of the Penn treebank.

5 Multi-Word Units

In another effort to improve generator accuracy over the baseline model we explored the use of multi-word units in generation. We expect that the identification of MWUs may be useful in imposing word-order constraints and reducing the complexity of the generation task. Take, for example, the following

$$\left[\text{APP}\left[\begin{array}{ll}\text{ADJUNCT} & \left[\begin{array}{ll}\text{PRED} & \text{`New'}\\ \text{NUM} & \text{sg}\\ \text{PERS} & 3\end{array}\right]\\ \text{PRED `York'} & \\ \text{NUM sg} & \\ \text{PERS 3} & \end{array}\right]\right]\left[\text{APP}\left[\begin{array}{ll}\text{PRED} & \text{`New\_York'}\\ \text{NUM} & \text{sg}\\ \text{PERS} & 3\end{array}\right]\right]\left[\text{APP}\left[\begin{array}{ll}\text{ADJUNCT} & \left[\begin{array}{ll}\text{PRED} & \text{`New'/NE1\_1}\\ \text{NUM} & \text{sg}\\ \text{PERS} & 3\end{array}\right]\\ \text{PRED `York'/NE1\_2} & \\ \text{NUM sg} & \\ \text{PERS 3} & \end{array}\right]\right]$$

Figure 5: Three different f-structure formats. From left to right: the original f-structure format; the MWU chunk format; the MWU mark-up format.

two sentences which show the gold version of a sentence followed by the version of the sentence produced by the generator:

Gold *By this time , it was 4:30 a.m. in New York , and Mr. Smith fielded a call from a New York customer wanting an opinion on the British stock market , which had been having troubles of its own even before Friday 's New York market break .*

Test *By this time , in New York , it was 4:30 a.m. , and Mr. Smith fielded a call from* **New** *a customer* **York** *, wanting an opinion on the market British stock which had been having troubles of its own even before Friday 's New York market break .*

The gold version of the sentence contains a multi-word unit, *New York*, which appears fragmented in the generator output. If multi-word units were either treated as one token throughout the generation process, or, alternatively, if a constraint were imposed on the generator such that multi-word units were always generated in the correct order, then this should help improve generation accuracy. In Section 5.1 we describe the various techniques that were used to incorporate multi-word units into the generation process and in 5.2 we detail the different types and sources of multi-word unit used in the experiments. Section 6 provides evaluation results on test and development sets from the WSJ treebank.

5.1 Incorporating MWUs into the Generation Process

We carried out three types of experiment which, in different ways, enabled the generation process to respect the restrictions on word-order provided by multi-word units. For the first experiments (type 1), the WSJ treebank *training* and *test* data were altered so that multi-word units are concatenated into single words (for example, *New York* becomes *New_York*). As in (Cahill and van Genabith, 2006) f-structures are generated from the (now altered) treebank and from this data, along with the treebank trees, the PCFG-based grammar, which is used for training the generation model, is extracted. Similarly, the f-structures for the test and development sets are created from Penn Treebank trees which have been modified so that multi-word units form single units. The leftmost and middle f-structures in Figure 5 show an example of an original f-structure format and a named-entity chunked format, respectively. Strings output by the generator are then post-processed so that the concatenated word sequences are converted back into single words.

In the second experiment (type 2) only the *test* data was altered with no concatenation of MWUs carried out on the training data.

In the final experiments (type 3), instead of concatenating named entities, a constraint is introduced to the generation algorithm which penalises the generation of sequences of words which violate the internal word order of named entities. The input is marked-up in such a way that, although named entities are no longer chunked together to form single words, the algorithm can read which items are part of named entities. See the rightmost f-structure in Figure 5 for an example of an f-structure marked-up in this way. The tag *NE1_1*, for example, indicates that the sub-f-structure is part of a named identity with id number 1 and that the item corresponds to the first word of the named entity. The baseline generation algorithm, following Kay (1996)'s work on chart generation, already contains the hard constraint that when combining two chart edges they must cover disjoint sets of words. We added an additional constraint which prevents edges from being combined if this would result in the generation of a string which contained a named entity which was either incomplete or where the words in the named entity were generated in the wrong order.

5.2 Types of MWUs used in Experiments

We carry out experiments with multi-word units from three different sources. First, we use the output of the maximum entropy-based named entity recognition system of (Chieu and Ng, 2003). This system identifies four types of named entity: person, organisation, location, and miscellaneous. Additionally we use a dictionary of candidate multi-word expressions based on a list from the Stanford Multiword Expression Project[4]. Finally, we also carry out experiments with multi-word units extracted from the BBN Pronoun Coreference and Entity Type Corpus (Weischedel and Brunstein, 2005). This supplements the Penn WSJ treebank's one million words of syntax-annotated Wall Street Journal text with additional annotations of 23 named entity types, including nominal-type named entities such as person, organisation, location, etc. as well as numeric types such as date, time, quantity and money. Since the BBN corpus data is very comprehensive and is hand-annotated we take this be be a gold standard, representing an upper bound for any gains that might be made by identifying complex named entities in our experiments.[5] Table 8 gives examples of the various types of MWUs identified by the three sources.

For our purposes we are not concerned with the distinctions between different types of named entities; we are merely exploiting the fact that they may be treated as atomic units in the generation model. In all cases we disregard multi-word units that cross the original syntactic bracketing of the WSJ treebank. An overview of the various types of multi-word units used in our experiments is presented in Table 9.

6 Experimental Evaluation

All experiments were carried out on the WSJ treebank with sections 02-21 for training, section 24 for development and section 23 for final test results. The LFG annotation algorithm of (Cahill et al., 2004) was used to produce the f-structures for development, test and training sets.

MWU type	Examples
Names	*Martha Matthews* *Yoshio Hatakeyama*
Organisations	*Rolls-Royce Motor Cars Inc.* *Washington State University*
Locations	*New York City* *New Zealand*
Time expressions	*October 19th* *two years ago* *the 21st century*
Quantities	*$2.7 million to $3 million* *about 25 %* *60 mph*
Prepositional expressions	*in fact* *at the time* *on average*

Table 8: Examples of some of the types of MWU from the three different sources.

	average number	average length
(Chieu and Ng, 2003)	0.61	2.40
Stanford MWE Project	0.10	2.48
BBN Corpus	1.15	2.66

Table 9: Average number of MWUs per sentence and average MWU length in the WSJ treebank grouped by MWU source.

Table 10 shows the final results for section 23. For each test we present BLEU score results as well as String Edit Distance and coverage. We measure statistical significance using two different tests. First we use a bootstrap resampling method, popular for machine translation evaluations, to measure the significance of improvements in BLEU scores, with a resampling rate of 1000.[6] We also calculated the significance of an increase in String Edit Distance by carrying out a paired t-test on the mean difference of the String Edit Distance scores. In Table 10, $\gg$ means significant at level 0.005. $>$ means significant at level 0.05.

In Table 10, *Baseline* gives the results of the generation algorithm of (Cahill and van Genabith, 2006). *HB Model* refers to the improved model with the increased history context, as described in Section 4. The results, where for example the BLEU score rises from 66.52 to 67.24, show that even increasing the conditioning context by a limited

[4] mwe.stanford.edu

[5] Although it is possible there are other types of MWUs that may be more suitable to the task than the named entities identified by BBN, so further gains might be possible.

[6] Scripts for running the bootstrapping method carried out in our evaluation are available for download at projectile.is.cs.cmu.edu/research/public/tools/bootStrap/tutorial.htm

	Section 23 (2416 sentences)				
Model	BLEU	StringEd	Coverage	BLEU Bootstrap Signif	StringEd Paired T-Test
1. Baseline	66.52	68.69	98.18		
2. HB Model	67.24	69.89	99.88	$\gg 1$	$\gg 1$
3. +MWU Best Automatic	67.81	70.36	99.92	$\gg 2$	$\gg 2$
4. MWU BBN	68.82	70.92	99.96	$\gg 3$	> 3

Table 10: Results on Section 23 for all sentence lengths.

amount increases the accuracy of the system significantly for both BLEU and String Edit Distance. In addition, coverage goes up from 98.18% to 99.88%.

+MWU Best Automatic displays our best results using automatically identified named entities. These were achieved using experiment type 2, described in Section 5, with the MWUs produced by (Chieu and Ng, 2003). Results displayed in Table 10 up to this point are cumulative. The final row in Table 10, *MWU BBN*, shows the best results with BBN MWUs: the history-based model with BBN multi-word units incorporated in a type 1 experiment.

We now discuss the various MWU experiments in more detail. See Table 11 for a breakdown of the MWU experiment results on the development set, WSJ section 24. Our baseline for these experiments is the history-based generator presented in Section 4. For each experiment type described in Section 5.1 we ran three experiments, varying the source of MWUs. First, MWUs came from the automatic NE recogniser of (Chieu and Ng, 2003), then we added the MWUs from the Stanford list and finally we ran tests with MWUs extracted from the BBN corpus.

Our first set of experiments (type 1), where both training data and development set data were MWU-chunked, produced the worst results for the automatically chunked MWUs. BLEU score accuracy actually decreased for the automatically chunked MWU experiments. In an error analysis of type 1 experiments with (Chieu and Ng, 2003) concatenated MWUs, we inspected those sentences where accuracy had decreased from the baseline. We found that for over half (51.5%) of these sentences, the input f-structures contained no multi-word units at all. The problem for these sentences therefore lay with the probabilistic grammar extracted from the MWU-chunked training data. When the source of MWU for the type 1 experiments was the BBN, however, accuracy improved significantly over the baseline and the result is the highest accuracy achieved over all experiment types. One possible reason for the low accuracy scores in the type 1 experiments with the (Chieu and Ng, 2003) MWU chunked data could be noisy MWUs which negatively affect the grammar. For example, the named entity recogniser of (Chieu and Ng, 2003) achieves an accuracy of 88.3% on section 23 of the Penn Treebank.

In order to avoid changing the grammar through concatenation of MWU components (as in experiment type 1) and thus risking side-effects which cause some heretofore likely constructions become less likely and vice versa, we ran the next set of experiments (type 2) which leave the original grammar intact and alter the input f-structures only. These experiments were more successful overall and we achieved an improvement over the baseline for both BLEU and String Edit Distance scores with all MWU types. As can be seen from Table 11 the best score for automatically chunked MWUs are with the (Chieu and Ng, 2003) MWUs. Accuracy decreases marginally when we added the Stanford MWUs. In our final set of experiments (type 3) although the accuracy for all three types of MWUs improves over the baseline, accuracy is a little below the type 2 experiments.

It is difficult to compare sentence generators since the information contained in the input varies greatly between systems, systems are evaluated on different test sets and coverage also varies considerably. In order to compare our system with those of (Nakanishi et al., 2005) and (Langkilde-Geary, 2002) we report our best results with automatically acquired MWUs for sentences of ≤ 20 words in length on section 23: our system gets coverage of 100% and a BLEU score of 71.39. For the same test set Nakanishi et al. (2005) achieved coverage of 90.75 and a BLEU score of 77.33. Langkilde-Geary (2002) re-

		Section 24 (1346 sentences)		
Model	MWUs	BLEU	StringEd	Coverage
HB Model		65.85	69.93	99.93
type 1	(Chieu and Ng, 2003)	65.81	70.34	99.93
(training and test data chunked)	+Stanford MWEs	64.81	69.67	99.93
	BBN	67.24	71.46	99.93
type 2	(Chieu and Ng, 2003)	66.37	70.26	99.93
(test data chunked)	+Stanford MWEs	66.28	70.21	99.93
	BBN	66.84	70.74	99.93
type 3	(Chieu and Ng, 2003)	66.30	70.12	100
(internal generation constraint)	+Stanford MWEs	66.07	70.02	99.93
	BBN	66.45	70.14	99.93

Table 11: Results on Section 24, all sentence lengths.

ports 82.7% coverage and a BLEU score of 75.7% on the same test set with the 'permute,no dir' type input. Langkilde-Geary (2002) report results for experiments with varying levels of linguistic detail in the input given to the generator. As with Nakanishi et al. (2005) we find the 'permute,no dir' type of input is most comparable to the level of information contained in our input f-structures. Finally, the symbolic generator of Callaway (2003) reports a Simple String Accuracy score of 88.84 and coverage of 98.7% on section 23 for all sentence lengths.

7 Conclusion and Future Work

We have presented techniques which improve the accuracy of an already state-of-art surface generation model. We found that a history-based model that increases conditioning context in PCFG style rules by simply including the grammatical function of the f-structure parent, improves generator accuracy. In the future we will experiment with increasing conditioning context further and using more sophisticated smoothing techniques to avoid sparse data problems when conditioning is increased.

We have also demonstrated that automatically acquired multi-word units can bring about moderate, but significant, improvements in generator accuracy. For automatically acquired MWUs, we found that this could best be achieved by concatenating input items when generating the f-structure input to the generator, while training the input generation grammar on the original (i.e. non-MWU concatenated) sections of the treebank. Relying on the BBN corpus as a source of multi-word units, we gave an upper bound to the potential usefulness of multi-word units in generation and showed that automatically acquired multi-word units, encouragingly, give results not far below the upper bound.

References

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th COLING*.

Anja Belz. 2007. Probabilistic generation of wether forecast texts. In *Proceedings of NAACL-HLT*.

Ezra Black, Fred Jelinek, John Lafferty, David M. Magerman, Robert Mercer, and Salim Roukos. 1992. Towards history-based grammars: Using richer models for probabilistic parsing. In *Proceeding of the 5th DARPA Speech and Language Workshop*.

Aoife Cahill and Josef van Genabith. 2006. Robust PCFG-based generation using automatically acquired LFG approximations. In *Proceedings of the 44th ACL*.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of the 42nd ACL*.

Charles B. Callaway. 2003. Evaluating coverage for large symbolic NLG grammars. In *In Proceedings of the 18th IJCAI*.

John A. Carroll and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Proceedings of IJCNLP*.

Eugene Charniak. 2000. A maximum entropy-inspired parser. In *Proceedings of the 1st NAACL*.

Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings of the CoNLL*.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Kevin Humphreys, Mike Calcagno, and David Weise. 2001. Reusing a statistical language model for generation. In *Proceedings of the 8th European Workshop on Natural Language Generation (EWNLG)*.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24.

Ronald M. Kaplan and Tracy Holloway King. 2003. Low-level mark-up and large-scale LFG grammar processing. In *Proceedings of the Lexical Functional Grammar Conference*.

Ron Kaplan. 1995. The formal architecture of lexical-functional grammar. In Dalrymple, Kaplan, Maxwell, and Zaenen, editors, *Formal Issues in Lexical-Functional Grammar*, pages 7–27. CSLI Publications.

Martin Kay. 1996. Chart generation. In *Proceedings of the 34th ACL*.

Irene Langkilde and Kevin Knight. 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 (ACL-COLING)*.

Irene Langkilde-Geary. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the 2nd INLG*.

Irene Langkilde. 2000. Forest-based statistical sentence generation. In *Proceedings of the 1st NAACL*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Hiroko Nakanishi, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proceedings of the 9th IWPT*.

Joakim Nivre and Jens Nilsson. 2004. Multiword units in syntactic parsing. In *Workshop on Methodologies and Evaluation of Multiword Units in Real-World Applications*.

Adwait Ratnaparkhi. 2000. Trainable methods for surface natural language generation. In *Proceedings of the 1st NAACL*.

Stefan Riezler and John T. Maxwell. 2006. Grammatical machine translation. In *Proceedings of the 6th NAACL*.

Stefan Riezler, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of the 3rd NAACL*.

Erik Velldal and Stephan Oepen. 2005. Maximum entropy models for realization ranking. In *Proceedings of the MT-Summit*.

Ralph Weischedel and Ada Brunstein, 2005. *BBN pronoun coreference and entity type corpus*. Technical Report.