

Robust PCFG-Based Generation using Automatically Acquired LFG Approximations

Aoife Cahill¹ and Josef van Genabith^{1,2}

¹ National Centre for Language Technology (NCLT)
School of Computing, Dublin City University, Dublin 9, Ireland

² Center for Advanced Studies, IBM Dublin, Ireland
{acahill, josef}@computing.dcu.ie

Abstract

We present a novel PCFG-based architecture for robust probabilistic generation based on wide-coverage LFG approximations (Cahill et al., 2004) automatically extracted from treebanks, maximising the probability of a tree given an f-structure. We evaluate our approach using string-based evaluation. We currently achieve coverage of 95.26%, a BLEU score of 0.7227 and string accuracy of 0.7476 on the Penn-II WSJ Section 23 sentences of length ≤ 20 .

1 Introduction

Wide coverage grammars automatically extracted from treebanks are a corner-stone technology in state-of-the-art probabilistic parsing. They achieve robustness and coverage at a fraction of the development cost of hand-crafted grammars. It is surprising to note that to date, such grammars do not usually figure in the complementary operation to parsing – natural language surface realisation.

Research on statistical natural language surface realisation has taken three broad forms, differing in where statistical information is applied in the generation process. Langkilde (2000), for example, uses n-gram word statistics to rank alternative output strings from symbolic hand-crafted generators to select paths in parse forest representations. Bangalore and Rambow (2000) use n-gram word sequence statistics in a TAG-based generation model to rank output strings and additional statistical and symbolic resources at intermediate generation stages. Ratnaparkhi (2000) uses maximum entropy models to drive generation with word bigram or dependency representations taking into account (unrealised) semantic features. Valldal and Oepen (2005) present a discriminative disambiguation model using a hand-crafted HPSG

grammar for generation. Belz (2005) describes a method for building statistical generation models using an automatically created *generation treebank* for weather forecasts. None of these probabilistic approaches to NLG uses a full treebank grammar to drive generation.

Bangalore et al. (2001) investigate the effect of training size on performance while using grammars automatically extracted from the Penn-II Treebank (Marcus et al., 1994) for generation. Using an automatically extracted XTAG grammar, they achieve a string accuracy of 0.749 on their test set. Nakanishi et al. (2005) present probabilistic models for a chart generator using a HPSG grammar acquired from the Penn-II Treebank (the Enju HPSG). They investigate discriminative disambiguation models following Valldal and Oepen (2005) and their best model achieves coverage of 90.56% and a BLEU score of 0.7723 on Penn-II WSJ Section 23 sentences of length ≤ 20 .

In this paper we present a novel PCFG-based architecture for probabilistic generation based on wide-coverage, robust Lexical Functional Grammar (LFG) approximations automatically extracted from treebanks (Cahill et al., 2004). In Section 2 we briefly describe LFG (Kaplan and Bresnan, 1982). Section 3 presents our generation architecture. Section 4 presents evaluation results on the Penn-II WSJ Section 23 test set using string-based metrics. Section 5 compares our approach with alternative approaches in the literature. Section 6 concludes and outlines further research.

2 Lexical Functional Grammar

Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982) is a constraint-based theory of grammar. It (minimally) posits two levels of representation, c(onstituent)-structure and f(unctional)-structure. C-structure is represented by context-free phrase-structure trees, and captures surface

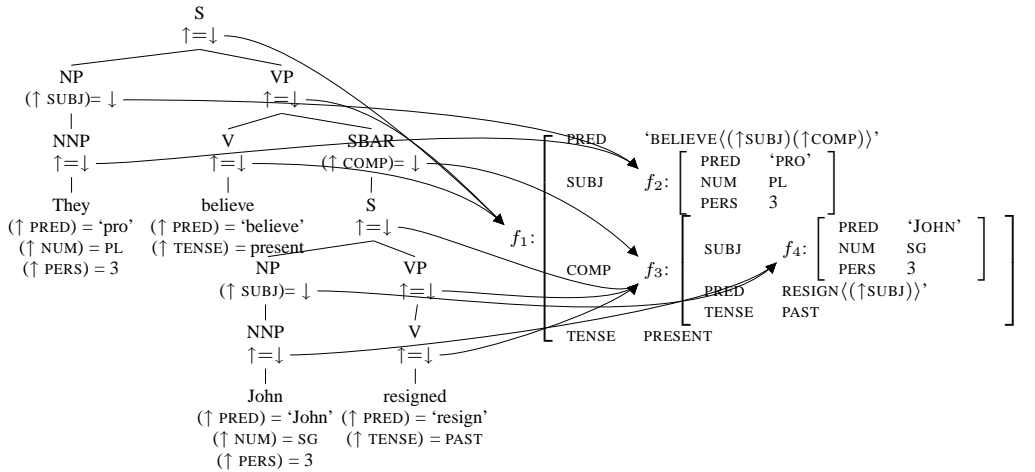


Figure 1: C- and f-structures for the sentence *They believe John resigned*.

grammatical configurations such as word order. The nodes in the trees are annotated with functional equations (attribute-value structure constraints) which are resolved to produce an f-structure. F-structures are recursive attribute-value matrices, representing abstract syntactic functions. F-structures approximate to basic predicate-argument-adjunct structures or dependency relations. Figure 1 shows the c- and f-structures for the sentence “*They believe John resigned*”.

3 PCFG-Based Generation for Treebank-Based LFG Resources

Cahill et al. (2004) present a method to automatically acquire wide-coverage robust probabilistic LFG approximations¹ from treebanks. The method is based on an automatic f-structure annotation algorithm that associates nodes in treebank trees with f-structure equations. For each tree, the equations are collected and passed on to a constraint solver which produces an f-structure for the tree. Cahill et al. (2004) present two parsing architectures: the **pipeline** and the **integrated** parsing architecture. In the **pipeline** architecture, a PCFG (or a history-based lexicalised generative parser) is extracted from the treebank and used to parse unseen text into trees, the resulting trees are annotated with f-structure equations by the f-structure annotation algorithm and a constraint solver produces an f-structure. In the **in-**

¹The resources are approximations in that (i) they do not enforce LFG completeness and coherence constraints and (ii) PCFG-based models can only approximate LFG and similar constraint-based formalisms (Abney, 1997).

egrated architecture, first the treebank trees are automatically annotated with f-structure information, f-structure annotated PCFGs with rules of the form $NP(\uparrow OBJ = \downarrow) \rightarrow DT(\uparrow = \downarrow) NN(\uparrow = \downarrow)$ are extracted, syntactic categories followed by equations are treated as monadic CFG categories during grammar extraction and parsing, unseen text is parsed into trees with f-structure annotations, the annotations are collected and a constraint solver produces an f-structure.

The generation architecture presented here builds on the integrated parsing architecture resources of Cahill et al. (2004). The generation process takes an f-structure (such as the f-structure on the right in Figure 1) as input and outputs the most likely f-structure annotated tree (such as the tree on the left in Figure 1) given the input f-structure

$$\operatorname{argmax}_{Tree} P(Tree | F-Str)$$

where the probability of a tree given an f-structure is decomposed as the product of the probabilities of all f-structure annotated productions contributing to the tree but where in addition to conditioning on the LHS of the production (as in the integrated parsing architecture of Cahill et al. (2004)) each production $X \rightarrow Y$ is now also conditioned on the set of f-structure features $Feats$ ϕ -linked² to the LHS of the rule. For an f-structure annotated tree $Tree$ and f-structure $F-Str$ with $\Phi(Tree) = F-Str$:³

² ϕ links LFG’s c-structure to f-structure in terms of many-to-one functions from tree nodes into f-structure.

³ Φ resolves the equations in $Tree$ into $F-Str$ (if satisfiable) in terms of the piece-wise function ϕ .

Conditioning F-Structure Features	Grammar Rules	Probability
{PRED, SUBJ, COMP, TENSE}	VP($\uparrow=\downarrow$) \rightarrow VBD($\uparrow=\downarrow$) SBAR(\uparrow COMP= \downarrow)	0.4998
{PRED, SUBJ, COMP, TENSE}	VP($\uparrow=\downarrow$) \rightarrow VBP($\uparrow=\downarrow$) SBAR(\uparrow COMP= \downarrow)	0.0366
{PRED, SUBJ, COMP, TENSE}	VP($\uparrow=\downarrow$) \rightarrow VBD($\uparrow=\downarrow$) , S(\uparrow COMP= \downarrow)	6.48e-6
{PRED, SUBJ, COMP, TENSE}	VP($\uparrow=\downarrow$) \rightarrow VBD($\uparrow=\downarrow$) S(\uparrow COMP= \downarrow)	3.88e-6
{PRED, SUBJ, COMP, TENSE}	VP($\uparrow=\downarrow$) \rightarrow VBP($\uparrow=\downarrow$) , SBARQ(\uparrow COMP= \downarrow)	7.86e-7
{PRED, SUBJ, COMP, TENSE}	VP($\uparrow=\downarrow$) \rightarrow VBD($\uparrow=\downarrow$) SBARQ(\uparrow COMP= \downarrow)	1.59e-7

Table 1: Example VP Generation rules automatically extracted from Sections 02–21 of the Penn-II Treebank

$$P(Tree|F-Str) := \prod_{\substack{X \rightarrow Y \text{ in Tree} \\ \phi(X) = Feats}} P(X \rightarrow Y|X, Feats) \quad (1)$$

$$P(X \rightarrow Y|X, Feats) = \frac{P(X \rightarrow Y, X, Feats)}{P(X, Feats)} = \quad (2)$$

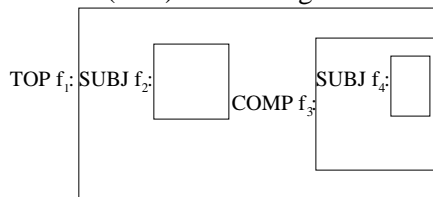
$$\frac{P(X \rightarrow Y, Feats)}{P(X, Feats)} \approx \frac{\#(X \rightarrow Y, Feats)}{\#(X \rightarrow \dots, Feats)} \quad (3)$$

and where probabilities are estimated using a simple MLE and rule counts (#) from the automatically f-structure annotated treebank resource of Cahill et al. (2004). Lexical rules (rules expanding preterminals) are conditioned on the full set of (atomic) feature-value pairs ϕ -linked to the RHS. The intuition for conditioning rules in this way is that local f-structure components of the input f-structure drive the generation process. This conditioning effectively turns the f-structure annotated PCFGs of Cahill et al. (2004) into probabilistic generation grammars. For example, in Figure 1 (where ϕ -links are represented as arrows), we automatically extract the rule $S(\uparrow=\downarrow) \rightarrow NP(\uparrow$ SUBJ= \downarrow) VP($\uparrow=\downarrow$) conditioned on the feature set {PRED,SUBJ,COMP,TENSE}. The probability of the rule is then calculated by counting the number of occurrences of that rule (and the associated set of features), divided by the number of occurrences of rules with the same LHS and set of features. Table 1 gives example VP rule expansions with their probabilities when we train a grammar from Sections 02–21 of the Penn Treebank.

3.1 Chart Generation Algorithm

The generation algorithm is based on chart generation as first introduced by Kay (1996) with Viterbi-pruning. The generation grammar is first converted into Chomsky Normal Form (CNF). We recursively build a chart-like data structure in a bottom-up fashion. In contrast to packing of locally equivalent edges (Carroll and Oepen, 2005),

in our approach if two chart items have equivalent rule left-hand sides and lexical coverage, only the most probable one is kept. Each grammatical function-labelled (sub-)f-structure in the overall f-structure indexes a (sub-)chart. The chart for each f-structure generates the most probable tree for that f-structure, given the internal set of conditioning f-structure features and its grammatical function label. At each level, grammatical function indexed charts are initially unordered. Charts are linearised by generation grammar rules once the charts themselves have produced the most probable tree for the chart. Our example in Figure 1 generates the following grammatical function indexed, embedded and (at each level of embedding) unordered (sub-)chart configuration:



For each local subchart, the following algorithm is applied:

```

Add lexical rules
While subchart is Changing
  Apply unary productions
  Apply binary productions
Propagate compatible rules

```

3.2 A Worked Example

As an example, we step through the construction of the COMP-indexed chart at level f_3 of the f-structure in Figure 1. For lexical rules, we check the feature set at the sub-f-structure level and the values of the features. Only features associated with lexical material are considered. The SUBJ-indexed sub-chart f_4 is constructed by first adding the rule $NNP(\uparrow=\downarrow) \rightarrow John(\uparrow$ PRED='John', \uparrow NUM=pl, \uparrow PERS=3). If more than one lexical rule corresponds to a particular set of features and values in the f-structure, we add all rules with different LHS categories. If two or more

rules with equal LHS categories match the feature set, we only add the most probable one.

Unary productions are applied if the RHS of the unary production matches the LHS of an item already in the chart *and* the feature set of the unary production matches the conditioning feature set of the local sub-f-structure. In our example, this results in the rule $\text{NP}(\uparrow\text{SUBJ}=\downarrow) \rightarrow \text{NNP}(\uparrow=\downarrow)$, conditioned on $\{\text{NUM}, \text{PERS}, \text{PRED}\}$, being added to the sub-chart at level f_4 (the probability associated with this item is the probability of the rule multiplied by the probability of the previous chart item which combines with the new rule). When a rule is added to the chart, it is automatically associated with the yield of the rule, allowing us to propagate chunks of generated material upwards in the chart. If two items in the chart have the same LHS (and the same yield independent of word order), only the item with the highest probability is kept. This Viterbi-style pruning ensures that processing is efficient.

At sub-chart f_4 there are no binary rules that can be applied. At this stage, it is not possible to add any more items to the sub-chart, therefore we propagate items in the chart that are compatible with the sub-chart index SUBJ. In our example, only the rule $\text{NP}(\uparrow\text{SUBJ}=\downarrow) \rightarrow \text{NNP}(\uparrow=\downarrow)$ (which yields the string *John*) is propagated to the next level up in the overall chart for consideration in the next iteration. If the yield of an item being propagated upwards in the chart is subsumed by an element already at that level, the subsumed item is removed. This results in efficiently treating the well known problem originally described in Kay (1996), where one unnecessarily retains sub-optimal strings. For example, generating the string “*The very tall strong athletic man*”, one does not want to keep variations such as “*The very tall man*”, or “*The athletic man*”, if one can generate the entire string. Our method ensures that only the most probable tree with the longest yield will be propagated upwards.

The COMP-indexed chart at level f_3 of the f-structure is constructed in a similar fashion. First the lexical rule $\text{V}(\uparrow=\downarrow) \rightarrow \text{resigned}$ is added. Next, conditioning on $\{\text{PRED}, \text{SUBJ}, \text{TENSE}\}$, the unary rule $\text{VP}(\uparrow=\downarrow) \rightarrow \text{V}(\uparrow=\downarrow)$ (with yield *resigned*) is added. We combine the new $\text{VP}(\uparrow=\downarrow)$ rule with the $\text{NP}(\uparrow\text{SUBJ}=\downarrow)$ already present from the previous iteration to enable us to add the rule $\text{S}(\uparrow=\downarrow) \rightarrow \text{NP}(\uparrow\text{SUBJ}=\downarrow) \text{VP}(\uparrow=\downarrow)$, conditioned

on $\{\text{PRED}, \text{SUBJ}, \text{TENSE}\}$. The yield of this rule is *John resigned*. Next, conditioning on the same feature set, we add the rule $\text{SBAR}(\uparrow\text{comp}=\downarrow) \rightarrow \text{S}(\uparrow=\downarrow)$ with yield *John resigned* to the chart. It is not possible to add any more new rules, so at this stage, only the $\text{SBAR}(\uparrow\text{COMP}=\downarrow)$ rule with yield *John resigned* is propagated up to the next level.

The process continues until at the outermost level of the f-structure, there are no more rules to be added to the chart. At this stage, we search for the most probable rule with TOP as its LHS category and return the yield of this rule as the output of the generation process. Generation fails if there is no rule with LHS TOP at this level in the chart.

3.3 Lexical Smoothing

Currently, the only smoothing in the system applies at the lexical level. Our backoff uses the built-in lexical macros⁴ of the automatic f-structure annotation algorithm of Cahill et al. (2004) to identify potential part-of-speech categories corresponding to a particular set of features. Following Baayen and Sproat (1996) we assume that unknown words have a probability distribution similar to hapax legomena. We add a lexical rule for each POS tag that corresponds to the f-structure features at that level to the chart with a probability computed from the original POS tag probability distribution multiplied by a very small constant. This means that lexical rules seen during training have a much higher probability than lexical rules added during the smoothing phase. Lexical smoothing has the advantage of boosting coverage (as shown in Tables 3, 4, 5 and 6 below) but slightly degrades the quality of the strings generated. We believe that the tradeoff in terms of quality is worth the increase in coverage.

Smoothing is *not* carried out when there is no suitable *phrasal* grammar rule that applies during the process of generation. This can lead to the generation of partial strings, since some f-structure components may fail to generate a corresponding string. In such cases, generation outputs the concatenation of the strings generated by the remaining components.

4 Experiments

We train our system on WSJ Sections 02–21 of the Penn-II Treebank and evaluate against the raw

⁴The lexical macros associate POS tags with sets of features, for example the tag NNS (plural noun) is associated with the features $\uparrow\text{PRED}=\$LEMMMA$ and $\uparrow\text{NUM}=\text{pl}$.

S. length	≤ 20	≤ 25	≤ 30	≤ 40	all
Training	16667	23597	29647	36765	39832
Test	1034	1464	1812	2245	2416

Table 2: Number of training and test sentences per sentence length

strings from Section 23. We use Section 22 as our development set. As part of our evaluation, we experiment with sentences of varying length (20, 25, 30, 40, all), both in training and testing. Table 2 gives the number of training and test sentences for each sentence length. In each case, we use the automatically generated f-structures from Cahill et al. (2004) from the original Section 23 treebank trees as f-structure input to our generation experiments. We automatically mark adjunct and coordination scope in the input f-structure. Notice that these automatically generated f-structures are not “perfect”, i.e. they are not guaranteed to be complete and coherent (Kaplan and Bresnan, 1982): a local f-structure may contain material that is not supposed to be there (incoherence) and/or may be missing material that is supposed to be there (incompleteness). The results presented below show that our method is robust with respect to the quality of the f-structure input and will always attempt to generate partial output rather than fail. We consider this an important property as pristine generation input cannot always be guaranteed in realistic application scenarios, such as probabilistic transfer-based machine translation where generation input may contain a certain amount of noise.

4.1 Pre-Training Treebank Transformations

During the development of the generation system, we carried out error analysis on our development set WSJ Section 22 of the Penn-II Treebank. We identified some initial pre-training transformations to the treebank that help generation.

Punctuation: Punctuation is not usually encoded in f-structure representations. Because our architecture is completely driven by rules conditioned by f-structure information automatically extracted from an f-structure annotated treebank, its placement of punctuation is not principled. This led to anomalies such as full stops appearing mid sentence and quotation marks appearing in undesired locations. One partial solution to this was to reduce the amount of punctuation that the system trained on. We removed all punctuation

apart from commas and full stops from the training data. We did not remove any punctuation from the evaluation test set (Section 23), but our system will ever only produce commas and full stops. In the evaluation (Tables 3, 4, 5 and 6) we are penalised for the missing punctuation. To solve the problem of full stops appearing mid sentence, we carry out a punctuation post-processing step on all generated strings. This removes mid-sentence full stops and adds missing full stops at the end of generated sentences prior to evaluation. We are working on a more appropriate solution allowing the system to generate all punctuation.

Case: English does not have much case marking, and for parsing no special treatment was encoded. However, when generating, it is very important that the first person singular pronoun is *I* in the nominative case and *me* in the accusative. Given the original grammar used in parsing, our generation system was not able to distinguish nominative from accusative contexts. The solution we implemented was to carry out a grammar transformation in a pre-processing step, to automatically annotate personal pronouns with their case information. This resulted in phrasal and lexical rules such as $NP(\uparrow SUBJ) \rightarrow PRP^{nom}(\uparrow = \downarrow)$ and $PRP^{nom}(\uparrow = \downarrow) \rightarrow I$ and greatly improved the accuracy of the pronouns generated.

4.2 String-Based Evaluation

We evaluate the output of our generation system against the raw strings of Section 23 using the Simple String Accuracy and BLEU (Papineni et al., 2002) evaluation metrics. Simple String Accuracy is based on the string edit distance between the output of the generation system and the gold standard sentence. BLEU is the weighted average of n-gram precision against the gold standard sentences. We also measure coverage as the percentage of input f-structures that generate a string. For evaluation, we automatically expand all contracted words. We only evaluate strings produced by the system (similar to Nakanishi et al. (2005)).

We conduct a total of four experiments. The parameters we investigate are lexical smoothing (Section 3.3) and partial output. Partial output is a robustness feature for cases where a sub-f-structure component fails to generate a string and the system outputs a concatenation of the strings generated by the remaining components, rather than fail completely.

Sentence length of Training Data	Evaluation Metric	Section 23 Sentences of length:				
		≤ 20	≤ 25	≤ 30	≤ 40	all
≤ 20	BLEU	0.6812	0.6601	0.6373	0.6013	0.5793
	String Accuracy	0.7274	0.7052	0.6875	0.6572	0.6431
	Coverage	96.52	95.83	94.59	93.76	93.92
≤ 25	BLEU	0.6915	0.6800	0.6696	0.6396	0.6233
	String Accuracy	0.7262	0.7095	0.6983	0.6731	0.6618
	Coverage	96.52	95.83	94.59	93.76	93.92
≤ 30	BLEU	0.6979	0.6881	0.6792	0.6576	0.6445
	String Accuracy	0.7317	0.7169	0.7075	0.6853	0.6749
	Coverage	97.97	97.95	97.41	97.15	97.31
≤ 40	BLEU	0.7045	0.6951	0.6852	0.6715	0.6605
	String Accuracy	0.7349	0.7212	0.7074	0.6881	0.6788
	Coverage	98.45	98.36	98.01	97.82	97.93
all	BLEU	0.7077	0.6974	0.6859	0.6734	0.6651
	String Accuracy	0.7373	0.7221	0.7087	0.6894	0.6808
	Coverage	98.65	98.5	98.12	97.95	98.05

Table 3: Generation +partial output +lexical smoothing

Sentence length of Training Data	Evaluation Metric	Section 23 Sentences of length:				
		≤ 20	≤ 25	≤ 30	≤ 40	all
all	BLEU	0.6253	0.6097	0.5887	0.5730	0.5590
	String Accuracy	0.6886	0.6688	0.6513	0.6317	0.6207
	Coverage	91.20	91.19	90.84	90.33	90.11

Table 4: Generation +partial output -lexical smoothing

Varying the length of the sentences included in the training data (Tables 3 and 5) shows that results improve (both in terms of coverage and string quality) as the length of sentence included in the training data increases.

Tables 3 and 5 give the results for the experiments including lexical smoothing and varying partial output. Table 3 (+partial, +smoothing) shows that training on sentences of all lengths and evaluating all strings (including partial outputs), our system achieves coverage of 98.05%, a BLEU score of 0.6651 and string accuracy of 0.6808. Table 5 (-partial, +smoothing) shows that coverage drops to 89.49%, BLEU score increases to 0.6979 and string accuracy to 0.7012, when the system is trained on sentences of all lengths. Similarly, for strings ≤ 20 , coverage drops from 98.65% to 95.26%, BLEU increases from 0.7077 to 0.7227 and String Accuracy from 0.7373 to 0.7476. Including partial output increases coverage (by more than 8.5 percentage points for all sentences) and hence robustness while slightly decreasing quality.

Tables 3 (+partial, +smoothing) and 4 (+partial, -smoothing) give results for the experiments including partial output but varying lexical smoothing. With no lexical smoothing (Table 4), the system (trained on all sentence lengths) produces

strings for 90.11% of the input f-structures and achieves a BLEU score of 0.5590 and string accuracy of 0.6207. Switching off lexical smoothing has a negative effect on all evaluation metrics (coverage and quality), because many more strings produced are now partial (since for PRED values unseen during training, no lexical entries are added to the chart).

Comparing Tables 5 (-partial, +smoothing) and 6 (-partial, -smoothing), where the system does not produce any partial outputs and lexical smoothing is varied, shows that training on all sentence lengths, BLEU score increases from 0.6979 to 0.7147 and string accuracy increases from 0.7012 to 0.7192. At the same time, coverage drops dramatically from 89.49% (Table 5) to 47.60% (Table 6).

Comparing Tables 4 and 6 shows that while partial output almost doubles coverage, this comes at a price of a severe drop in quality (BLEU score drops from 0.7147 to 0.5590). On the other hand, comparing Tables 5 and 6 shows that lexical smoothing achieves a similar increase in coverage with only a very slight drop in quality.

5 Discussion

Nakanishi et al. (2005) achieve 90.56% coverage and a BLEU score of 0.7723 on Section 23

Sentence length of Training Data	Evaluation Metric	Section 23 Sentences of length:				
		≤ 20	≤ 25	≤ 30	≤ 40	all
≤ 20	BLEU	0.7326	0.7185	0.7165	0.7082	0.7052
	String Accuracy	0.76	0.7428	0.7363	0.722	0.7175
	Coverage	85.49	81.56	77.26	71.94	69.08
≤ 25	BLEU	0.7300	0.7235	0.7218	0.7118	0.7077
	String Accuracy	0.7517	0.7382	0.7315	0.7172	0.7116
	Coverage	89.65	87.77	84.38	80.31	78.56
≤ 30	BLEU	0.7207	0.7125	0.7107	0.6991	0.6946
	String Accuracy	0.747	0.7336	0.7275	0.711	0.7045
	Coverage	93.23	92.14	89.74	86.59	85.18
≤ 40	BLEU	0.7221	0.7140	0.7106	0.7016	0.6976
	String Accuracy	0.746	0.7331	0.7236	0.7072	0.7001
	Coverage	94.58	93.85	91.89	89.62	88.33
all	BLEU	0.7227	0.7145	0.7095	0.7011	0.6979
	String Accuracy	0.7476	0.7331	0.7239	0.7077	0.7012
	Coverage	95.26	94.40	92.55	90.69	89.49

Table 5: Generation -partial output +lexical smoothing

Sentence length of Training Data	Evaluation Metric	Section 23 Sentences of length:				
		≤ 20	≤ 25	≤ 30	≤ 40	all
all	BLEU	0.7272	0.7237	0.7201	0.7160	0.7147
	String Accuracy	0.7547	0.7436	0.7361	0.7237	0.7192
	Coverage	61.99	57.38	53.64	47.60	47.60

Table 6: Generation -partial output -lexical smoothing

sentences, restricted to length ≤ 20 for efficiency reasons. Langkilde-Geary’s (2002) best system achieves 82.8% coverage, a BLEU score of 0.924 and string accuracy of 0.945 against Section 23 sentences of all lengths. Callaway (2003) achieves 98.7% coverage and a string accuracy of 0.6607 on sentences of all lengths. Our best results for sentences of length ≤ 20 are coverage of 95.26%, BLEU score of 0.7227 and string accuracy of 0.7476. For all sentence lengths, our best results are coverage of 89.49%, a BLEU score of 0.6979 and string accuracy of 0.7012.

Using hand-crafted grammar-based generation systems (Langkilde-Geary, 2002; Callaway, 2003), it is possible to achieve very high results. However, hand-crafted systems are expensive to construct and not easily ported to new domains or other languages. Our methodology, on the other hand, is based on resources automatically acquired from treebanks and easily ported to new domains and languages, simply by retraining on suitable data. Recent work on the automatic acquisition of multilingual LFG resources from treebanks for Chinese, German and Spanish (Burke et al., 2004; Cahill et al., 2005; O’Donovan et al., 2005) has shown that given a suitable treebank, it is possible to automatically acquire high quality LFG re-

sources in a very short space of time. The generation architecture presented here is easily ported to those different languages and treebanks.

6 Conclusion and Further Work

We present a new architecture for stochastic LFG surface realisation using the automatically annotated treebanks and extracted PCFG-based LFG approximations of Cahill et al. (2004). Our model maximises the probability of a tree given an f-structure, supporting a simple and efficient implementation that scales to wide-coverage treebank-based resources. An improved model would maximise the probability of a string given an f-structure by summing over trees with the same yield. More research is required to implement such a model efficiently using packed representations (Carroll and Oepen, 2005). Simple PCFG-based models, while effective and computationally efficient, can only provide approximations to LFG and similar constraint-based formalisms (Abney, 1997). Research on discriminative disambiguation methods (Valldal and Oepen, 2005; Nakanishi et al., 2005) is important. Kaplan and Wedekind (2000) show that for certain linguistically interesting classes of LFG (and PATR etc.) grammars, generation from f-structures yields a context free language. Their proof involves the notion of a

“refinement” grammar where f-structure information is compiled into CFG rules. Our probabilistic generation grammars bear a conceptual similarity to Kaplan and Wedekind’s “refinement” grammars. It would be interesting to explore possible connections between the treebank-based empirical work presented here and the theoretical constructs in Kaplan and Wedekind’s proofs.

We presented a full set of generation experiments on varying sentence lengths training on Sections 02–21 of the Penn Treebank and evaluating on Section 23 strings. Sentences of length ≤ 20 achieve coverage of 95.26%, BLEU score of 0.7227 and string accuracy of 0.7476 against the raw Section 23 text. Sentences of all lengths achieve coverage of 89.49%, BLEU score of 0.6979 and string accuracy of 0.7012. Our method is robust and can cope with noise in the f-structure input to generation and will attempt to produce partial output rather than fail.

Acknowledgements

We gratefully acknowledge support from Science Foundation Ireland grant 04/BR/CS0370 for the research reported in this paper.

References

- Stephen Abney. 1997. Stochastic Attribute-Value Grammars. *Computational Linguistics*, **23**(4):597–618.
- Harald Baayen and Richard Sproat. 1996. Estimating lexical priors for low-frequency morphologically ambiguous forms. *Computational Linguistics*, **22**(2):155–166.
- Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of COLING 2000*, pages 42–48, Saarbrücken, Germany.
- Srinivas Bangalore, John Chen, and Owen Rambow. 2001. Impact of quality and quantity of corpora on stochastic generation. In *Proceedings of EMNLP 2001*, pages 159–166.
- Anja Belz. 2005. Statistical generation: Three methods compared and evaluated. In *Proceedings of the 10th European Workshop on Natural Language Generation (ENLG’ 05)*, pages 15–23, Aberdeen, Scotland.
- Michael Burke, Olivia Lam, Rowena Chan, Aoife Cahill, Ruth O’Donovan, Adams Bodom, Josef van Genabith, and Andy Way. 2004. Treebank-Based Acquisition of a Chinese Lexical-Functional Grammar. In *Proceedings of the 18th Pacific Asia Conference on Language, Information and Computation*, pages 161–172, Tokyo, Japan.
- 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 ACL-04*, pages 320–327, Barcelona, Spain.
- Aoife Cahill, Martin Forst, Michael Burke, Mairead McCarthy, Ruth O’Donovan, Christian Rohrer, Josef van Genabith, and Andy Way. 2005. Treebank-based acquisition of multilingual unification grammar resources. *Journal of Research on Language and Computation; Special Issue on “Shared Representations in Multilingual Grammar Engineering”*, pages 247–279.
- Charles B. Callaway. 2003. Evaluating coverage for large symbolic NLG grammars. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 811–817, Acapulco, Mexico.
- John Carroll and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Proceedings of IJCNLP05*, pages 165–176, Jeju Island, Korea.
- Ron Kaplan and Joan Bresnan. 1982. Lexical Functional Grammar, a Formal System for Grammatical Representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, MA.
- Ron Kaplan and Juergen Wedekind. 2000. LFG Generation produces Context-free languages. In *Proceedings of COLING 2000*, pages 141–148, Saarbrücken, Germany.
- Martin Kay. 1996. Chart Generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 200–204, Santa Cruz, CA.
- Irene Langkilde-Geary. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Second International Natural Language Generation Conference*, pages 17–24, Harriman, NY.
- Irene Langkilde. 2000. Forest-based statistical sentence generation. In *Proceedings of NAACL 2000*, pages 170–177, Seattle, WA.
- Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 110–115, Princeton, NJ.
- Hiroko Nakanishi, Yusuke Miyao, and Jun’ichi Tsujii. 2005. Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proceedings of the International Workshop on Parsing Technology*, Vancouver, Canada.
- Ruth O’Donovan, Aoife Cahill, Josef van Genabith, and Andy Way. 2005. Automatic Acquisition of Spanish LFG Resources from the CAST3LB Treebank. In *Proceedings of LFG 05*, pages 334–352, Bergen, Norway.
- Kishore Papineni, Salim Roukos, Todd Ward, and WeiJing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL 2002*, pages 311–318, Philadelphia, PA.
- Adwait Ratnaparkhi. 2000. Trainable methods for natural language generation. In *Proceedings of NAACL 2000*, pages 194–201, Seattle, WA.
- Erik Valldal and Stephan Oepen. 2005. Maximum Entropy Models for Realization Reranking. In *Proceedings of the 10th Machine Translation Summit*, pages 109–116, Phuket, Thailand.