Natural Language Engineering **17** (4): 455–483. © Cambridge University Press 2010
doi:10.1017/S1351324910000288

Dependency-based n-gram models for general purpose sentence realisation

YUQING GUO[1], HAIFENG WANG[2] and JOSEF VAN GENABITH[3]

[1]*Toshiba (China) Research and Development Center*
5/F., Tower W2, Oriental Plaza, Dongcheng District, Beijing, 100738, China
e-mail: guoyuqing@rdc.toshiba.com.cn
[2]*Baidu, Inc., Baidu Campus, No. 10, Shangdi 10th Street, Haidian District, Beijing, 100085, China*
e-mail: wanghaifeng@baidu.com
[3]*NCLT/CNGL, School of Computing, Dublin City University Glasnevin, Dublin 9, Ireland*
e-mail: josef@computing.dcu.ie

(*Received 29 March 2010; revised 15 September 2010; accepted 25 October 2010; first published online 29 November 2010*)

Abstract

This paper presents a general-purpose, wide-coverage, probabilistic sentence generator based on dependency n-gram models. This is particularly interesting as many semantic or abstract syntactic input specifications for sentence realisation can be represented as labelled bi-lexical dependencies or typed predicate-argument structures. Our generation method captures the mapping between semantic representations and surface forms by linearising a set of dependencies *directly*, rather than via the application of grammar rules as in more traditional chart-style or unification-based generators. In contrast to conventional n-gram language models over surface word forms, we exploit structural information and various linguistic features inherent in the dependency representations to constrain the generation space and improve the generation quality. A series of experiments shows that dependency-based n-gram models generalise well to different languages (English and Chinese) and representations (LFG and CoNLL). Compared with state-of-the-art generation systems, our general-purpose sentence realiser is highly competitive with the added advantages of being simple, fast, robust and accurate.

1 Introduction and overview

Sentence generation,[1] or more precisely, surface realisation is the final stage in the typical pipeline architecture for the natural language generation (NLG) systems (Reiter and Dale 1997). The aim of sentence realisation is to produce a syntactically, morphologically and orthographically correct sentence from a semantic or deep syntactic representation. This process potentially involves several subtasks: (i) determining the linear order of words and phrases; (ii) inserting necessary function words and punctuation and (iii) performing morphological inflections.

[1] In this paper, we use 'generation' as a cover term for what should otherwise be more strictly referred to as 'tactical generation' or 'surface realisation'.

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

Even though NLG has not been studied as well as its sibling – natural language understanding – there is now a considerable body of literature on sentence realisation, using techniques varying along a number of dimensions, including

- handcrafted grammars versus treebank-based automatically induced grammars;
- reversible versus non-reversible approaches;
- symbolic versus statistical models;
- grammar versus direct mapping approaches.

From a knowledge engineering-based perspective, a sentence realiser is a module encoding knowledge that defines the correspondence between natural language utterances and their meaning representations. In the traditional NLG systems, this knowledge is encoded in the form of natural language grammars manually constructed by linguistic experts following certain linguistic principles. Some of the most influential systems in this category include FUF (Elhadad 1993), a natural language surface generator that interacts with the Systemic Unification Realisation Grammar of English (SURGE) (Elhadad and Robin 1996) developed under the formalism of functional unification grammar (Kay 1979); Penman/KPML (Bateman 1997), a portable generator for domain-independent large-scale text generation, as well as an environment for developing multilingual grammars written within the framework of systemic-functional linguistics (Halliday 1978); and CoGenTex's RealPro (Lavoie and Rambow 1997), which converts deep syntactic structures into surface forms by a cascade of rules based on the Meaning-Text Theory (MTT) (Mel'čuk 1988: 43–101).[2]

The traditional symbolic grammar-based generators often hit a knowledge acquisition bottleneck: hand-crafting grammar rules are knowledge-intensive, time-consuming, difficult to scale to unrestricted text and can be prohibitively expensive to port them to new domains and languages. Over the last fifteen years, with the availability of an increasing number of treebanks, extensive efforts have been made to automatically acquire grammatical resources from treebanks for both parsing and generation. A number of wide-coverage grammars in rich formalisms, including Tree Adjoining Grammar (TAG), Head-Driven Phrase Structure Grammar (HPSG), Lexical-Functional Grammar (LFG) and Combinatory Categorial Grammar (CCG), have been automatically acquired from the Penn Treebank, and have been used in statistical sentence realisers (Bangalore and Rambow 2000; Nakanishi, Nakanishi and Tsujii 2005; Cahill and van Genabith 2006; Hogan *et al.* 2007; White *et al.* 2007). In most of these approaches, given generation input, surface syntactic representations are constructed using the treebank grammars and a variety of statistical approaches. Statistical techniques have also been explored in pruning edges in symbolic chart generation (White 2004), a sequence of corpus-trained classifiers have been used to incrementally transform semantic input into a fully articulated surface structure (Corston-Oliver *et al.* 2002; Marciniak and Strube

[2] More grammar-based natural language generation systems can be found in an extensive list of existing NLG systems at http://www.nlg-wiki.org/systems/

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

2004), and probabilistic generation grammars have been acquired automatically from unannotated data rather than ready-made treebanks (Zhong and Stent 2005; Belz 2007; DeVault, Traum and Artstein 2008).

Often, statistical approaches are used by sentence realisers in a generate-and-select paradigm. The first stage applies either handcrafted or automatically acquired grammar rules to generate a space of all possible paraphrases, while the second stage selects the most likely realisation(s) from the space by ranking the candidates in the order of their probabilities. By and large, the following two statistical models have been applied in the ranking: (i) n-gram language models over different units, such as word-level bigram/trigram models (Langkilde and Knight 1998; Langkilde 2000; Bangalore and Rambow 2000), or factored language models integrating word-level n-grams with n-grams over part-of-speech tags and category labels (White *et al.* 2007); (ii) log-linear models incorporating a wider variety of syntactic and semantic features (Nakanishi *et al.* 2005; Velldal and Oepen 2005; Cahill, Forst and Rohrer 2007).

Even though the ultimate purpose of sentence realisation is to generate well-formed surface strings, the great majority of (deterministic or probabilistic) grammar-based generation systems achieve their goal by building surface syntactic structures (primarily organised around phrase structure descriptions) compatible with given input semantic specifications, via application of grammar rules, and the final output string is then formed by reading words from the leaf nodes of the syntactic tree in (left-to-right) order (or the other way round). This 'detour' in the generation process incurs an overhead in the development and maintenance of a large number of grammar rules, as well as in the construction and management of complex syntactic structures during generation. More problematic with the generate-and-select approach is the fact that a form of 'overgeneration' is effectively required for producing the full set of realisation candidates, which is computationally expensive in practice.

A contrasting alternative is to map from semantic configurations to surface strings *directly*, side-stepping the construction of intermediate syntactic representations during generation. Some systems have made attempts in this direction, focusing on specialised applications or particular subproblems in sentence realisation: Ratnaparkhi (2000) construes the generation process as attribute ordering and lexical choice given semantic representations implemented as attribute-value pairs, training maximum entropy models using a corpus of generation templates in the air travel domain. A number of studies concentrate on the problem of constituent ordering within a local dependency or unordered phrase structure, utilising a range of linguistic features and maximum entropy classifiers (Uchimoto *et al.* 2000; Filippova and Strube 2007) or decision trees (Ringger *et al.* 2003). While this research work is inspiring and shows promising results, it has focused on particular linguistic phenomena and languages.

The research described in this paper extends the approaches described above to models for full-coverage direct surface realisation, building practical and generic sentence generators that are accurate, efficient, robust and reusable for a number of languages and input representations. Taking a statistical approach, our general-purpose realiser takes labelled dependency representations as input, and maps them

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

directly to surface strings without constructing intermediate constituent trees (or other syntactic representations). We design general Dependency-based N-gram (DN-gram)[3] probabilistic generation models that integrate structural information from the dependency representation to linearise dependencies directly. We evaluate DN-gram models on two typologically different languages – English and Chinese – as well as two input dependency representations – LFG f-structures and CoNLL dependencies.

The research reported in this paper builds on and substantially extends an early version of the DN-gram model reported in Guo, van Genabith and Wang (2008) in important respects: it uses log-linear model combinations (rather than simple linear interpolations) with substantially improved generation accuracy and provides the (to the best of our knowledge) first results for generation with CoNLL-style dependencies, a format popular in current state-of-the-art dependency parsing (McDonald, Crammer and Pereira 2005; Nivre 2006). Our best DN-gram model achieves 0.8065 BLEU, 0.7871 SSA and 31.54 per cent exact match on English Penn-II data using LFG f-structure input, 0.7713 BLEU, 0.7484 SSA and 32.89 per cent exact match for Chinese Penn Treebank data using LFG f-structure input, and 0.8820 BLEU, 0.8596 SSA and 47.76 per cent exact match for CoNLL-dependency input on English Penn-II data, all with 100 per cent coverage and average generation times of about 0.87 to 0.16 seconds per sentence for our English and Chinese data, respectively.

Section 2 gives an overview of the general dependency inputs and sketches the basic idea of the DN-gram generation model. Section 3 describes a number of increasingly sophisticated DN-gram models and presents the generation algorithm in detail. We describe the experiments and provide results across different languages and input representations in Section 4. We compare our results with previous work and discuss the outcomes in Section 5. Finally, we conclude with a summary and outline directions for the future research.

2 Preliminaries

2.1 Generation input

In principle, an abstract meaning representation is provided as input to a sentence realiser; in practice, the actual input varies widely from one system to the next because of the variability of representations and linguistic formalisms that a generation system is designed for. On the one hand, the input can be drawn from different linguistic levels of representation, accordingly varying in the degree of 'abstraction'. Certain systems, e.g. RealPro (Lavoie and Rambow 1997), adopt very low level of abstraction, including syntactic categories, lexical specifications, definiteness and tense decisions, etc. in the input specification; some systems, e.g. Penman/KPML (Bateman 1997), adopt high level of abstraction, including pure semantic roles, such as *actor*, *temporal-locating*, etc.; some, for example Nitrogen/HALogen (Langkilde

[3] We use the term 'DN-gram' to refer to the dependency-based structural n-grams to distinguish them from the conventional surface-string-oriented n-gram language model.

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

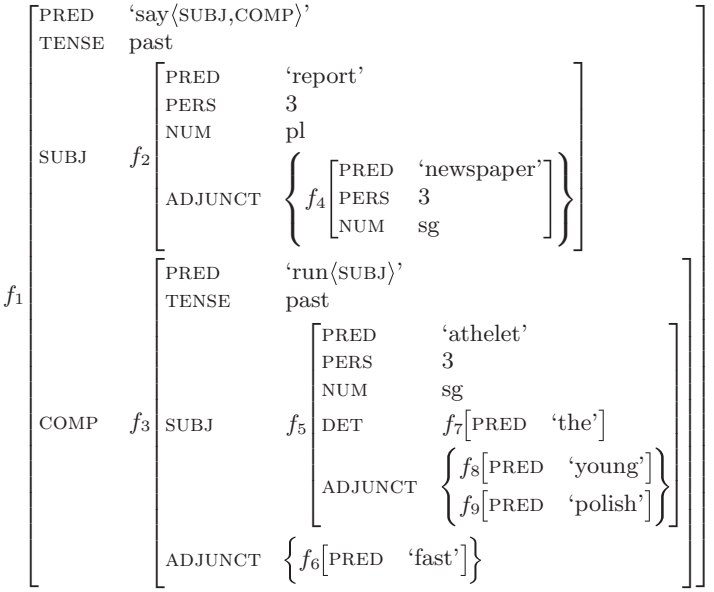
2002), take a wide variety of input ranging from rather abstract semantic representation to more fully specified syntactic representation. On the other hand, the expression of the semantic content may take different forms, such as Sentence Plan Language (SPL) specifications adopted by PENMAN and NITROGEN/HALOGEN, functional descriptions as inputs to FUF/SURGE (Elhadad and Robin 1996), typed feature structures representing logical forms for LINGO/LKB (Carroll *et al.* 1999) and the cognate treebank-based HPSG generator (Nakanishi *et al.* 2005), non-hierarchical conceptual graphs as semantic input specifications used by the PROTECTOR system (Nicolov and Mellish 2000), semantic dependency graphs as used in AMALGAM (Corston-Oliver *et al.* 2002) and OPENCCG (White 2004) and many others. Even though different input representations are specific with respect to their underlying linguistic theories, from an engineering perspective these can be characterised by a common representation in terms of attribute value matrices (AVMs), viz. a recursive structure consisting of attribute names and their values encoding linguistic information at various levels of abstraction. These AVMs, in turn, can be converted to dependency trees or graphs in computational implementations.

In this paper, we present a general-purpose sentence realiser for generic dependency-based input representations. We instantiate the framework using two types of dependency representations popular in the CL/NLP and linguistics literature: on the one hand we use the slightly more abstract Lexical-Functional Grammar (LFG)-based f-structure representations (Kaplan and Bresnan 1982; Dalrymple *et al.* 1995); on the other hand we use the more 'surfacy' dependency representations familiar from the CoNLL (2006–2009) shared tasks on dependency parsing. Below, we first concentrate on the LFG-based representations and turn to the CoNLL-style dependencies in Section 4.2.2.

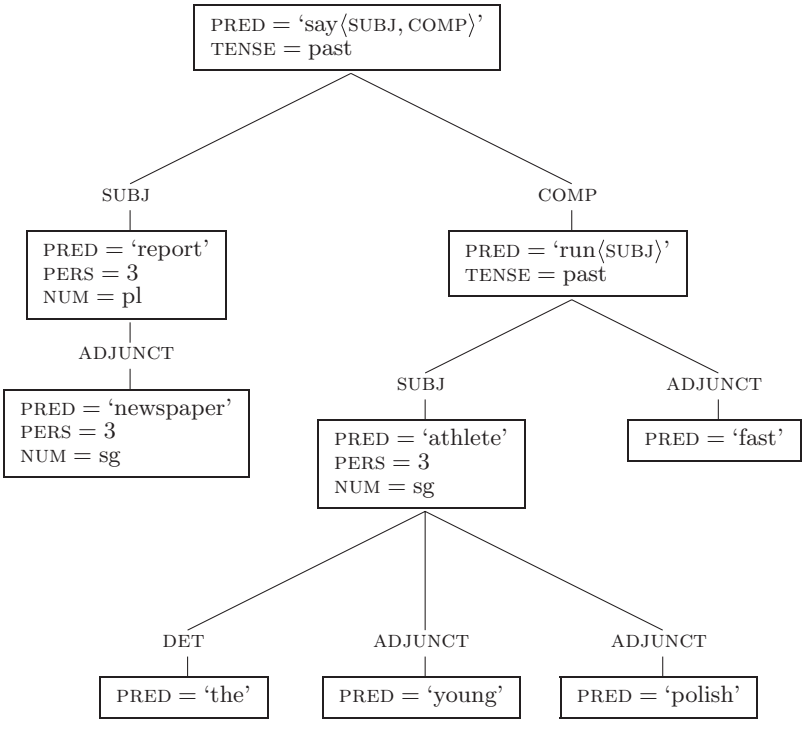
LFG is a constraint-based grammar formalism, which postulates (minimally) two levels of representation: c(onstituent)-structure and f(unctional)-structure. C-structures are conventional phrase structure trees encoding linear order, hierarchical groupings and syntactic categories of constituents, representing surface syntactic configurations for a particular language. F-structures encode more abstract functional relations, like SUBJ(ect), OBJ(ect) and ADJUNCT etc., providing deep syntactic analyses across languages. An f-structure is represented in the form of an AVM or a labelled dependency graph, as illustrated in Figure 1, and consists of the following types of attributes:

- PRED is the predicate/head of a local f-structure. The value of PRED is a lemma associated with a subcategorisation frame reflecting the unique semantic contribution of each word in the sentence.
- The atomic-valued features describe linguistic properties of the predicates, such as TENSE, MOOD, ASPECT, PERS, NUM, etc.
- Grammatical Functions (GFs) indicate the functional relationship between a predicate and the sub-f-structures/dependents. GFs come in the following two different types:
  - Arguments are subcategorised for by the local predicate, such as SUBJ(ect), OBJ(ect), OBL(ique) etc., and thus only occur once in each local f-structure.

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press



(a) the canonical representation of f-structure as AVM



(b) a representation of f-structure as dependency tree

Fig. 1. An example of the f-structure and the equivalent dependency tree for the sentence *Newspaper reports said the young Polish athlete ran fast.*

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

– Modifiers, like ADJUNCT, COORD(inate), etc., are not subcategorised for by the predicate, and can occur any number of times in a local f-structure.

An LFG-based generator takes as input an f-structure and determines the surface strings that correspond to the given f-structure (Kaplan and Wedekind 2000).

2.2 Generation from F-structures

A number of sentence realisers have been built based on LFG using either symbolic or statistical approaches. The ParGram/XLE system (Crouch *et al.* 2007) comes with a fully fledged unification-based generator and multilingual wide-coverage LFG grammars consisting of handcrafted syntactic rules and lexical entries. Based on the XLE resources, Cahill *et al.* (2007) describe a two-stage surface realiser for German, which first produces a generation space with the symbolic XLE generator, and then selects the best realisation using a log-linear ranker. Cahill and van Genabith (2006) and Hogan *et al.* (2007) present conditional PCFG generation models implemented in chart-style generators for English. Both systems are based on wide-coverage, probabilistic LFG approximations automatically acquired from the Penn-II treebank. A common characteristic of previous LFG-based generators is the use of bi-directional (i.e. reversible) grammar rules (either handcrafted or treebank-based), where the generation process can be viewed as the reverse of parsing. In a sense, generation is driven by an input f-structure representation, but proceeds through functionally annotated context-free grammar (CFG or PCFG) rules and a chart generator to construct the corresponding c-structure, from which final surface strings can be read. By contrast, this paper presents a simple dependency- and n-gram-based, probabilistic generation model, which cuts out the middle-man (i.e. the CFG-component) from previous approaches.

In accordance with a generic dependency formalism, an LFG f-structure is not linearly ordered, i.e. it does not carry information on to the relative surface order of the predicates in an f-structure. Furthermore, predicates in f-structure are lemmas (rather than surface forms), with morphological and semantic information relevant to surface realisation encoded as atomic-valued features. In order to generate a well-formed string from an f-structure, first and foremost we need to put in correct order the set of words corresponding to the predicate lemmas in the f-structure. This is typically performed by a word-based n-gram model operating on the yields of the trees compatible with the input f-structure. As mentioned above, the traditional language model in the (over-)generate-and-select approach is prohibitively expensive to apply, because vast numbers of alternative realisations are produced and word n-gram-based language model probabilities are calculated for each of them. For example, Langkilde (2000) reports that NITROGEN deals with trillions of alternatives in the course of generation. To reduce the number of possible realisations, we break down the generation space, as defined by the given f-structure, into each layer of embedding within the f-structure: that is to say, the realiser selects the most likely substring for each sub-f-structure, and concatenates the substrings into a complete sentence by recursively traversing and linearising each level of embedding in the f-structure, respecting linearisations of the already established sub-f-structures, until

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

it reaches the outermost f-structure. This reduction in complexity depends on the simplifying assumption that the input f-structure consists of projective dependencies only[4] (as in the example in Figure 1), such that each sub-f-structure can be realised independently of other sub-f-structures (other than, of course, the sub-f-structures that are part of the local sub-f-structure), as all dependencies are with respect to local heads and are single-headed. A similar scenario is obtained in dependency parsing, where most practical dependency parsers (McDonald *et al.* 2005; Nivre 2006) are restricted to projective structures. This is partially due to the fact that a majority of sentences in most languages can be represented by projective dependencies (Mel'čuk 1988). In the case of English, Surdeanu *et al.* (2008) report that in the complete corpus used in the CoNLL-2008 shared task on dependency parsing, only 0.4 per cent of dependencies are non-projective.

While our dependency-based realiser assumes projectivity, most theoretical formulations of dependency grammar (and in particular LFG) recognise the need for non-projective representations of certain linguistic constructions, e.g. long-distance dependencies (LDDs) or (more generally) non-local dependencies (NLDs). Figure 2 exemplifies an f-structure analysis of a sentence discussed in Carroll *et al.* (1999), where the non-local dependency is represented as a re-entrancy or coindexation in the f-structure and the dependency graph equivalent. The re-entrancy in the f-structure indicates that the word *how* is associated with two grammatical functions as well as two heads: it bears the discourse function FOCUS of the matrix clause, and at the same time also functions as an ADJUNCT modifying the predicate *run* in the embedded clause. Alternatively, an account of NLDs in earlier transformational grammar involves movement, viz. *how* is moved from the position at which it should be interpreted semantically, often known as a 'gap' or 'trace', to the location where it is realised, which is called a 'filler' or 'antecedent'. Although varying in analysis and terminology, a common property of these accounts is that the interrogative word *how* is interpreted to relate to two distinct sites – a semantic site (the 'trace') and a surface site (the 'antecedent'). The grammatical function ADJUNCT associated with the semantic site is non-projective because it is governed/dominated by the head verb *run* in the embedded dependency structure, but in the surface string *how* is realised at the initial position of sentence outside the substring of the embedded clause *the athlete ran*. By contrast, the discourse function FOCUS associated with the surface site is locally dependent on the matrix verb and forms a projective dependency. In our approach, we convert non-projective dependencies into projective dependencies: for each sentence realisation we need to maintain the surface site (i.e. the antecedent discourse function) of the NLD while removing the semantic site (i.e. the trace), effectively transforming dependency graphs into dependency trees. Therefore, conversion from non-projective to projective representations involves (i) identifying the non-local dependencies within the f-structure; (ii) removing the 'trace' (and associated GF) and keeping the 'antecedent', or, in other words, moving the trace up towards the antecedent site (as indicated by the dashed curves in

[4] A dependency tree is projective if, for every arc $h \rightarrow d$ in the tree and node w, w occurs between h and d in the surface word order only if w is dominated by h.

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

(1) How did the newspapers say the athlete ran?

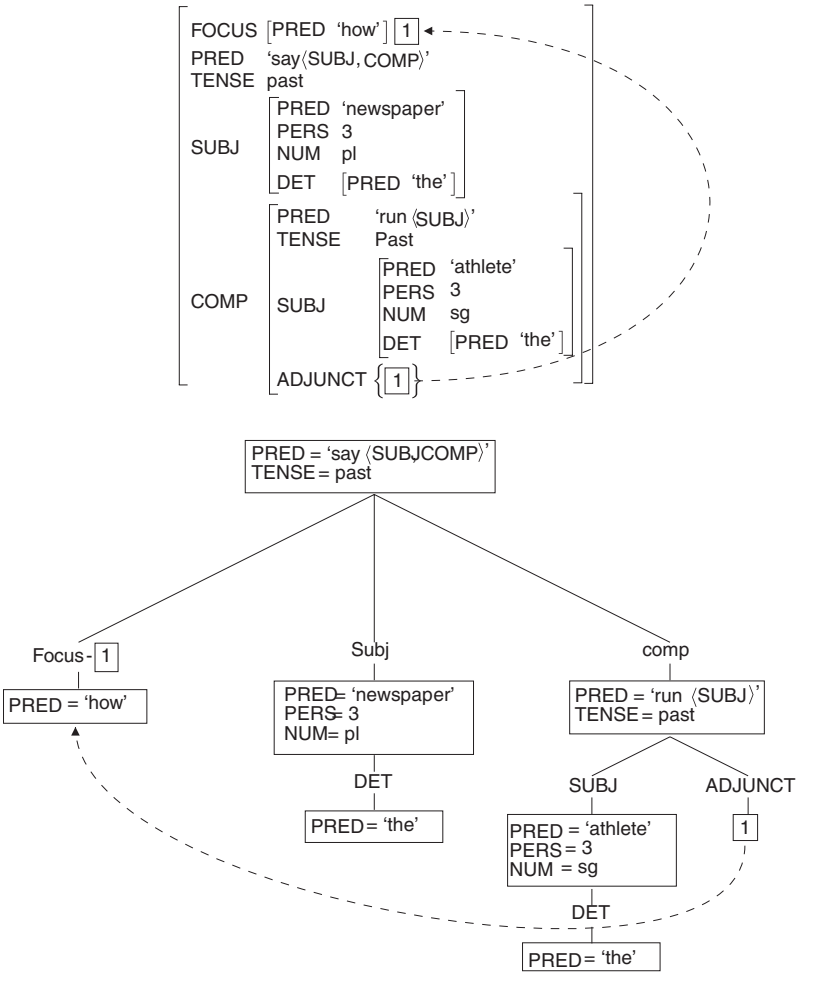


Fig. 2. LDD representations in f-structure and dependency tree.

Figure 2). In our LFG implementation for English, three grammaticalised discourse functions, FOCUS, TOPIC and TOPIC_REL, characterise NLDs arising in different types of questions, relative clauses and topicalisations, respectively. As a consequence, the conversion is directly triggered by the three discourse functions. The identification of non-local dependencies in other semantic representations or for other languages may require more elaborate linguistically motivated analyses, for example, Gamon *et al.* (2002) address extraposition of relative clauses in German.

3 Dependency-based surface realisation

3.1 Dependency-based n-gram models

Given an f-structure, the primary task for surface realisation is to determine the linear order of words or phrases that the f-structure generates. The n-gram models

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

are easy to build and use in practice, hence have become the most popular tool for word ordering in a variety of NLP tasks. However, a traditional language model relying exclusively on lexical information between adjacent words is unable to capture relationships between words spanning a longer distance, and usually performs worse on a clause level than on a phrase level (Filippova and Strube 2009). Furthermore, from a linguistic point of view, grammatical relations or semantic roles to a large extent govern the word order of sentences in languages such as English and Chinese. Based on these observations, in our model we perform linearisation at each local f-structure level by means of the DN-gram models, i.e. n-gram models over dependency (D) labels or grammatical functions (GFs), which describe languages in terms of functions equally on the phrase level and the clause level. For the example in Figure 1, the GF sequence DET $\prec$ ADJUNCT $\prec$ PRED realises the noun phrase *the young polish athlete* for the embedded f-structure f_5; in the same manner but at the embedding f-structure f_3, the GF sequence SUBJ $\prec$ PRED $\prec$ ADJUNCT produces the clause *the young polish athlete ran fast*, reusing and leaving intact the linearisation produced for the embedded f_5.

Formally, in our most basic model, for a local f-structure f_i containing m GFs (including PRED), generating a surface string $S_1^m = s_1 \ldots s_m$ expressed by f_i is equivalent to linearising all the GFs present at f_i. The DN-gram model calculates probabilities for all permutations $GF_1^m = GF_1 \ldots GF_m$, and searches for the best surface sequence that maximises the probability $P(S_1^m)$ in terms of maximising $P(GF_1^m)$. Applying the chain rule and the Markov assumption, the probability of the surface realisation is computed according to (1).

$$P(S_1^m) = P\left(GF_1^m\right) = P(GF_1 \ldots GF_m) = \prod_{k=1}^{m} P\left(GF_k | GF_{k-n+1}^{k-1}\right) \tag{1}$$

The basic DN-gram model over bare GFs is not a good probability estimator as it only makes use of a few dozen grammatical function roles. For example, there is no way to capture the difference between two ADJUNCT modifiers according to the labels of the two GFs. In order to facilitate better decisions, we extend the basic model to a number of more complex DN-gram models incorporating contextual and fine-grained lexical information. Equation (2) describes a model that additionally conditions GF sequences on the parent GF $Parent_i$ of the local f-structure f_i being linearised and (3) conditions the n-gram probability on the headword (or predicate) $Head_i$ governing f_i,

$$P^p\left(S_1^m\right) = \prod_{k=1}^{m} P\left(GF_k | GF_{k-n+1}^{k-1}, Parent_i\right) \tag{2}$$

$$P^h\left(S_1^m\right) = \prod_{k=1}^{m} P\left(GF_k | GF_{k-n+1}^{k-1}, Head_i\right) \tag{3}$$

Alternatively, we can also integrate additional features with the GFs being ordered. Equation (4) stands for a featured model, where $Feat_k$ stands for the function label GF_k augmented with the set of atomic-valued features that specify the sub-f-structure assigned to the attribute GF_k. In contrast with common atomic features included in

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

Table 1. *Examples of different DN-gram models for f_3 in Figure 1*

Model	DN-gram unit		
gf (P)	SUBJ	PRED	ADJUNCT
parent (P^p)	SUBJ-COMP	PRED-COMP	ADJUNCT-COMP
head (P^h)	SUBJ-'run'	PRED-'run'	ADJUNCT-'run'
feat (P^f)	SUBJ[PERS=3,NUM=sg]	PRED[TENSE=past]	ADJUNCT[]
lex (P^l)	SUBJ[PRED='athlete']	PRED[PRED='run']	ADJUNCT[PRED='fast']

the feature model, e.g. TENSE, MOOD and NUM, which have a closed (i.e. finite) set of values, the value set of PRED is given by a (theoretically) unlimited number of lemmas corresponding to lexical entries. We separate out PRED from other atomic-valued features and develop a fully lexicalised DN-gram model in (5), where Lex_k stands for GF_k augmented with its predicate lemma,

$$P^f\left(S_1^m\right) = \prod_{k=1}^{m} P\left(Feat_k | Feat_{k-n+1}^{k-1}\right) \tag{4}$$

$$P^l\left(S_1^m\right) = \prod_{k=1}^{m} P\left(Lex_k | Lex_{k-n+1}^{k-1}\right) \tag{5}$$

Table 1 exemplifies DN-gram models in (1) to (5) for realising the string *the young polish athlete ran fast* represented by f-structure f_3 in Figure 1. Other features from a larger context or at different granularity have the potential to be included in the DN-gram models. In this paper, however, we focus on the general methodology without a full exploration of all possible feature options.

In building the DN-gram models, we apply Good-Turing smoothing when calculating probabilities for unlexicalised (*gf*, *parent* and *feat*) DN-grams and interpolated Kneser–Ney smoothing for lexicalised (*head* and *lex*) DN-grams.

The individual models described from (1) to (5) can, in turn, be combined in different ways. As lexicalised models are likely to suffer from severe data sparseness, we create combined models by interpolating the individual DN-gram models incrementally (rather than by using chain rules). Klakow (1998) shows that, in general, log-linear interpolation (LLI) performs better than simple linear interpolation when combining n-gram models of different kinds. Accordingly, we implement LLIs to combine the estimates from different DN-gram models as follows:[5]

$$P^{LLI}\left(S_1^m\right) = \prod_i P_i\left(S_1^m\right)^{\lambda_i} \tag{6}$$

Here λ_i is the interpolation weight that adjusts the contribution of the component models P_i. In contrast to simple linear interpolation, no explicit constraints appear on the LLI weights. To estimate optimal sets of interpolation weights, we employ

[5] Formally, the result should be normalised over the sum of all words to ensure a proper probability distribution. Since the final goal is to select the best sentence realisation, we omit the normalising denominator from the formula.

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

(2) We continue *to* feel *that* the stock market will stabilise quickly.

$$
\begin{bmatrix}
\text{PRED} & \text{`continue'} \\
\text{TENSE} & \text{pres} \\
\text{NUM} & \text{pl} \\
\text{SUBJ} & f_2\begin{bmatrix} \text{PRED} & \text{`pro'} \\ \text{PERS} & 1 \\ \text{NUM} & \text{pl} \end{bmatrix} \\
f_1 & \begin{bmatrix}
\text{PRED} & \text{`feel'} \\
\text{TO_INF} & + \\
\text{XCOMP} & f_3\begin{bmatrix}
\text{PRED} & \text{`stabilise'} \\
\text{THAT} & + \\
\text{TENSE} & \text{fut} \\
\text{COMP} & f_4\begin{bmatrix}
\text{SUBJ} & f_5\begin{bmatrix}
\text{PRED} & \text{`market'} \\
\text{PERS} & 3 \\
\text{NUM} & \text{sg} \\
\text{ADJUNCT} & \left\{ f_6\begin{bmatrix} \text{PRED} & \text{`stock'} \\ \text{PERS} & 3 \\ \text{NUM} & \text{sg} \end{bmatrix} \right\} \\
\text{DET} & f_7[\text{PRED} \quad \text{`the'}]
\end{bmatrix} \\
\text{ADJUNCT} & \{ f_8[\text{PRED} \quad \text{`quickly'}] \}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

the minimum error rate training (MERT) algorithm (Och 2003), which has been widely adopted in statistical machine translation systems.

3.2 The realisation algorithm

In addition to word ordering, surface realisation also involves inserting function words and punctuation marks to produce a grammatical sentence.

In our LFG f-structure representations, function words are encoded in atomic-valued features at the local head level. From a linguistic perspective, the number of function words is limited and most of them occupy specific positions in sentences. For instance, the verbal particle *to* precedes the base form of a verb to form an infinitive in English, and the interrogative particle 吗 appears at the end of a question in Chinese etc. In our generation system, a small rule-based component captures heuristics about the use of function words: function words are triggered by a limited set of atomic-valued features and inserted in one of the four possible positions with respect to a local f-structure f_i: (i) before the local predicate PRED_i governing f_i; (ii) after the local predicate PRED_i; (iii) before the complete string s_i that is yield of f_i; and (iv) after the string s_i. For example, the f-structure attribute TO_INF indicates that the local predicate verb is infinitive and hence a particle *to* should be placed before it. As illustrated by sentence (2) and the f-structure that produces (2), *to* is generated immediately before the string position in (2) that realises the local predicate *feel* from f_3. Similarly, the subordinating conjunction *that* is used to introduce a subordinate clause (*the stock market* ...), as f_4 contains the feature-value pair THAT=+.

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

Table 2. *Examples of lexical macros*

Lexical macro	Surface form
pred='say', tense=past	said
pred='report', num=pl, pers=3	reports
pred='newspaper', num=sg, pers=3	newspaper
pred='run', tense=past	ran
pred='athlete', num=sg, pers=3	athlete

As a language-specific syntactic element, unlike in the CoNLL dependency representations (discussed in Section 4.2.2), punctuation is not conventionally represented at the level of LFG f-structures. We present a maximum entropy model to recover punctuation marks for unpunctuated realisation strings in a separate post-processing component (Guo, Wang and van Genabith 2010). However, this paper concentrates more on the overall dependency-based realisation scheme, and, for simplicity (and following the CoNLL dependency representations), we record all punctuation marks in the f-structure representations as the value of a PUNC attribute and generate them in the just same way as any other GF.

In order to generate orthographically and semantically correct sentences, morphological alternation has to be determined for inflecting languages, such as English.[6] Inflection indicates grammatical properties, such as number, person, case, or tense, which are encoded as atomic-valued features in f-structures. We call a lexeme together with its morphological features a *lexical macro*. A lexical macro, as exemplified in Table 2, maps a lexeme to the appropriate word form in the surface realisation. We learn the set of lexical macros automatically from our training corpora consisting of f-structures and the corresponding surface strings.

In summary, given an input f-structure F consisting of projective dependencies, the generation algorithm recursively traverses F in a bottom-up fashion, and at each level of embedding with (sub-) f-structure f_i:

(1) Instantiates the local predicate PRED_i at f_i and performs morphological inflections if necessary.
(2) Calculates DN-gram probabilities of possible GF permutations licensed by f_i.
(3) Finds the most probable GF sequence among all possibilities by Viterbi search.
(4) Generates the surface string s_i according to the best GF sequence as a realisation of f_i.
(5) Inserts function words in appropriate positions of s_i by heuristic rules and propagates s_i up to the next outer embedding f-structure.

4 Experiments and evaluation

To test the performance and coverage of the general DN-gram-based realisation model, we carry out experiments on two typologically different languages (English

[6] Realisation for uninflected languages, such as Chinese, does not require morphological variation.

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

and Chinese) and two different dependency representations (LFG f-structures and CoNLL dependencies).

4.1 Experiment design

Similar to many others (Corston-Oliver *et al.* 2002; Langkilde 2002; Callaway 2003), we use a pre-processing procedure to convert surface syntax trees, viz. constituent-structure trees hand coded in a treebank or automatically produced by a parser, to the semantic dependency representations expected by the surface realiser. In our experiments, the English Penn Treebank II (PTB-II) (Marcus, Santorini and Marcinkiewicz 1993) and the Chinese Penn Treebank (CTB) (Xue *et al.* 2005) are used as source data for testing and evaluation. We follow the methodologies developed in Cahill *et al.* (2004) and Guo, van Genabith and Wang (2007) to automatically convert the phrase structure treebank (or parser output) trees into f-structure/dependency banks for English and Chinese, respectively. Grammatical functions in f-structures are unordered. For the purpose of training DN-gram models, we automatically associate surface string positions (numerical word offset from start of the sentence) with the corresponding f-structure predicates to linearise the unordered set of GFs, and GF sequences are collected from each level of embedding in local f-structures for training. Unlike the training data, the f-structures input to the surface realiser do *not* contain any string position information. But, due to the particulars of the automatic f-structure annotation algorithm, the order of sub-f-structures in set-valued GFs, such as ADJUNCT and COORD, happens to correspond to their surface order. To avoid unfairly inflating evaluation results, we lexically reorder the GFs in each local f-structure before the generation process. This resembles the 'permute, no dir'-type experiment in Langkilde (2002), where the modifiers in the input feature-value structures are not ordered.

Experiments for English are carried out on the *Wall Street Journal* (WSJ) portion of the PTB-II, using standard training/test/development splits. That is, 39,832 sentences from sections 02–21 are used for training, 2,416 sentences from section 23 for testing and 1,700 sentences from section 22 are held out for development. The latest version of the Penn Chinese Treebank 6.0 (CTB6), excluding the portion of ACE broadcast news, is used for experiments on Chinese data. Following the recommended splits (in the list-of-file of the CTB6), we divide the data into 756 files with a total of 15,663 sentences as a training set, 84 files with 1,900 sentences as a test set and 50 files with 1,245 sentences as a development set. Table 3 gives an overview of all the data sets used in the experiments.[7]

In the experiments, we create DN-gram models with the SRILM toolkit (Stolcke 2002). It has been shown that more sophisticated n-gram models (e.g. with higher

[7] It is perhaps surprising to note that, on average, Chinese has longer sentences but slightly fewer f-structures per sentence, while it has the same number of grammatical functions per sub-f-structure compared to English. The main reason is that function words are represented indirectly as atomic valued features (and not as grammatical functions) in f-structures, and Chinese has about 1.72 function words per sentence while English has about 0.54 (all computed on the training sets).

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

Table 3. *English and Chinese data sets*

English	Sents[a]	SentLen[b]	F-strs[c]	F/S[d]	FstrLen[e]
Training	39,832	23.85	549,074	13.78	3.00
Test	2,416	23.46	32,137	13.30	3.00
Develop	1,700	23.00	23,323	13.72	2.99
Chinese	Sents[a]	SentLen[b]	F-strs[c]	F/S[d]	FstrLen[e]
Training	15,663	26.77	189,148	12.08	3.01
Test	1,900	28.31	24,412	12.85	3.01
Develop	1,245	28.59	16,179	13.00	3.01

[a] Total number of sentences.
[b] Mean number of tokens per sentence, including punctuation marks.
[c] Total number of sub-f-structures.
[d] Mean number of sub-f-structures per sentence.
[e] Mean number of GFs per sub-f-structure, including punctuation GFs.

n, complex smoothing techniques, clustering or caching) are often not worth the effort of implementing them due to data sparseness and other issues (Goodman 2001); therefore in the experiments reported below we implement trigram models throughout.

4.2 Experimental results

Following Langkilde (2002) and much other research on wide-coverage surface realisation, we evaluate the sentence generator by comparing the automatically produced strings to the original treebank sentences, and assess the generation performance in term of three accuracy metrics.

Exact match is the percentage of generated sentences that exactly match a corresponding reference sentence.

BLEU score (Papineni *et al.* 2002) is computed as a geometric average of n-gram precisions p_n of the output string with respect to the reference, using n-grams up to length N and uniform weights w_n summing to one. Equation (7) gives the formula, where BP is a sentence brevity penalty factor that adjusts the average n-gram precision, c is the length of the generated string and r is the length of the reference string. In our experiments, we set $N = 4$ as default:

$$BLEU = \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \times BP \tag{7}$$

$$BP = \begin{cases} \exp(1 - r/c) & \text{if } c \leq r \\ 1 & \text{if } c > r \end{cases}$$

$$\text{where} \quad w_n = 1/N$$

SSA stands for average NIST simple string accuracy, which reflects the average number of insertion (I), deletion (D) and substitution (S) errors between the

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

Table 4. *Results for LFG dependencies for English Penn-II WSJ section 23 and Chinese CTB6 test sets*

		English			Chinese		
	Model	ExMatch	BLEU	SSA	ExMatch	BLEU	SSA
1.	lang	6.26%	0.4587	0.3951	13.58%	0.4327	0.4659
2.	gf	7.08%	0.5456	0.5812	13.26%	0.5743	0.5351
3.	+parent	8.44%	0.5779	0.6121	14.84%	0.6106	0.5575
4.	+head	9.73%	0.5982	0.6211	15.74%	0.6232	0.5654
5.	+feat	11.55%	0.6294	0.6402	18.74%	0.6552	0.5970
6.	+lex	26.37%	0.7813	0.7606	25.68%	0.7105	0.6534
7.	+lex+head	26.82%	0.7815	0.7654	27.63%	0.7241	0.6628
8.	+lex+parent	28.68%	0.7923	0.7780	27.63%	0.7255	0.6620
9.	+lex+feat	29.26%	0.7977	0.7779	28.95%	0.7392	0.6742
10.	+lex+feat+head	29.55%	0.7992	0.7826	29.95%	0.7477	0.6787
11.	+lex+feat+parent	30.09%	0.8020	0.7826	30.00%	0.7500	0.6784
12.	all	30.30%	0.8034	0.7866	30.42%	0.7525	0.6797
13.	all+lang	**31.54%**	**0.8065**	**0.7871**	**32.89%**	**0.7713**	**0.7484**

reference strings (R) and the strings produced by the realiser. SSA is calculated as follows:

$$SSA = 1 - \frac{I + D + S}{R} \tag{8}$$

In contrast to many other systems, the dependency-based realisation algorithm guarantees that all input f-structures can produce a complete sentence, hence coverage-dependent evaluation (as adopted in most grammar-based generation systems) is not necessary in the following experiments.

4.2.1 Different languages

We carry out experiments on two typologically different languages: English and Chinese. Tables 4 provides the generation results for the standard English[8] and Chinese test sets. To set up a baseline, we first apply a conventional language model (*lang*) where only surface word forms are taken into account. To avoid prohibitive overgeneration occurring in previous language model-based approaches, generation proceeds under the projectivity assumption, and at each local f-structure only the realisation with the highest probability is propagated to the next higher level for further concatenation. As shown in the first line of Table 4, a word-based language model achieves only very modest accuracy for both English and Chinese. The DN-gram models starting with the basic model (gf) over bare dependency labels and moving on to models incorporating larger context or more fine-grained lexical

[8] Since capitalisation is ignored in the current generation system, all English characters are converted to lowercase for training and testing.

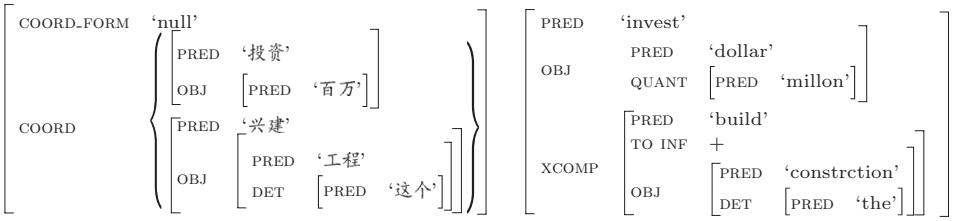
https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

and atomic-valued features are reported in lines 2–6. To avoid data sparseness, all extended DN-gram models are smoothed by interpolating them with the basic gf model using the log-linear interpolation described in (6) in Section 3. The first observation is that even the basic gf model makes a substantial difference compared to the *lang* model with respect to BLEU and SSA scores. And in turn, the models conditioned on parent GF (*parent*) and head word (*head*), and the models combined with syntactic features ($feat$) and lexical features (*lex*) clearly perform better than the basic gf model. In line with similar observations in parsing (where a lexicalised parser often outperforms a non-lexicalised one), the *lex* model stands out among all the generation models, boosting the BLEU score of the gf model by nearly 24 percentage points (absolute) for English and 14 percentage points (absolute) for Chinese. In the following, we adopt the *lex* model as a new basis and combine it gradually with some of the other feature models. This results in six combination models (lines 7–12) with increasing complexity. As can be expected, combining more models to include different types of features shows a steady increase in all evaluation metrics, although the magnitude of the improvement becomes smaller due to some overlap between the features. All the DN-gram models capture some kind of relationship between the nodes of the dependency structure, whereas the language model captures relationships between adjoining words in a sentence. We finally combine these two different types of n-gram models to see if further improvement can be achieved. We find that the role of the language model in English and Chinese is somewhat different. The language model appears to have a strong effect on Chinese, showing a significant improvement across the board, especially with respect to the SSA score. Although the addition of the language model also increases exact match for English by more than 1 percentage point, the improvements in BLEU and SSA scores for English are not that prominent. This difference can probably be explained by the fact that English is much richer than Chinese with respect to word forms. Morphological inflection, abbreviation (e.g. Calif. versus California) and different orthographic conventions (e.g. realise versus realize) are common in English but almost absent in Chinese. We conjecture that using a larger corpus to train the language model and performing more elaborate preprocessing to capture word variations should increase the effect of the language model on the English data.

Apart from the effects due to the language model, in general the DN-gram models perform similarly on the English and Chinese data. Nevertheless, comparing the most complex *all* model with the simplest gf model, we see that the improvement achieved by mixing more dependency features is evidently greater for English (with an increase of 23 exact match points, 26 BLEU points and 20 SSA points absolute)[9] than for Chinese (with an increase of 17 exact match points, 18 BLEU points and 14 SSA points absolute). This is likely to be the case as the English training set is more than twice the size of the Chinese training set. Apart from that, it probably reflects important differences between the two languages and data sets. Examining the development sets, we find that there are a total of 5,103 coordinated elements

[9] 1 point = 0.01 change in the respective score.

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press



(a) The Chinese f-structure (b) The English f-structure

Fig. 3. Different f-structure analyses for the sentence *投资百万兴建这个工程 /invest million dollars to build the construction.*

(4.10 per sentence) in Chinese, in contrast to 2,104 (1.24 per sentence) in English. Many coordinated elements are tied in the DN-gram probabilities and hence can not be correctly ordered during realisation, which to some extent accounts for the lower SSA scores for Chinese. To be specific, we distinguish the following three types of coordinations in Chinese:

- Coordinated on the syntactic level, but not on the semantic level, for example:
  (2) 投资 百万 兴建 这个 工程
  invest million build this construction
  'invest million dollars *to* build the construction'
- Coordinated on syntactic and semantic levels, but the phrase is usually expressed in a fixed order, for example:
  (3) 改革 开放
  reform opening-up
  'reform and opening up'
- Coordinated on syntactic and semantic levels, and the order of the two conjuncts can be swopped freely, for example:
  (4) (a) **充沛的精力和敏捷的思维**
  plentiful energy and quick thinking
  'energetic and agile'

  (b) 敏捷的 思维 和 充沛的 精力
  'agile and energetic'

Due to lack of inflection and case markers, many sequences of Chinese verb phrases are treated as conjuncts in the f-structure analyses, which give rise to the first type of coordinations, while, in many cases, their English counterparts can be distinguished into different grammatical functions. For example, the infinitive *to* in sentence (2) indicates that the second verb phrase functions as a complement, as illustrated in Figure 3(b). In principle, this type of syntactic coordination should be solved by a more precise semantic analysis on the f-structure level.

The other two types, as exemplified by (3) and (4), are both genuine semantic coordinations, but differ in the flexibility of their word order. In the former case, it is adequate to capture the word order of a fixed expression by a conventional language model, which has been partially reflected in the scores of the complex

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

Table 5. *Results for CoNLL dependencies for English Penn-II WSJ section 23*

	Treebank converted trees			Parser output		
Model	ExMatch	BLEU	SSA	ExMatch	BLEU	SSA
lang	7.37%	0.4558	0.3878	7.37%	0.4508	0.3859
gf	7.08%	0.5393	0.5583	6.75%	0.5174	0.5242
+head	7.49%	0.5622	0.5774	7.24%	0.5470	0.5490
+parent	8.40%	0.5670	0.5975	7.45%	0.5402	0.5574
+lex	34.52%	0.8253	0.8003	29.68%	0.7937	0.7516
+feat	37.83%	0.8431	0.8175	31.08%	0.8054	0.7779
all	46.40%	0.8777	0.8491	39.24%	0.8462	0.8186
all+lang	**47.76%**	**0.8820**	**0.8596**	**42.51%**	**0.8594**	**0.8423**

all+lang model for Chinese as listed in Table 4. In the latter case where the meaning can be expressed more freely, (4-a) & (4-b) are almost equally acceptable, and choosing any one of them would have the same effect on the actual generation quality (although it may not be reflected in the string-matching-based automatic evaluation results, if only one reference is given).

4.2.2 Different representations

Even though the DN-gram models are presented with LFG f-structures, these are general-purpose models and should be suitable for any labelled bilexical dependency or typed predicate-argument representations. To test this claim, we carry out a set of experiments on the dependency structures used in the CoNLL shared tasks (2006–2009) on dependency parsing.[10] We convert the Penn-II treebank trees into an English dependency bank, using the *pennconverter* toolkit[11] (Johansson and Nugues 2007) that produces dependency trees following the linguistic conventions of the CoNLL-2008 shared task. In a more realistic scenario, such as transfer-based machine translation, the input representations to a realisation component are likely to contain some noise and will not be as perfect as the dependency structures converted from the gold treebank trees. To approximate noisy data, we parse the PTB development and test sentences by a state-of-the-art dependency parser – the MSTParser (McDonald *et al.* 2005), and take the parsing output dependency trees as alternative realisation inputs. Table 5 shows the results for the various n-gram generation models given the representations obtained via treebank conversion and automatic parsing, respectively. To give a general picture of the sentences realised by the DN-gram models, Table 6 lists a set of example outputs (and their BLEU scores) for a sentence extracted from the Penn-II WSJ section 22 development set under the various generation models in Table 5, given the converted CoNLL dependencies as inputs.

[10] http://ifarm.nl/signll/conll/
[11] http://nlp.cs.lth.se/software/treebank_converter/

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

Table 6. *Example outputs given by CoNLL dependencies*

Model	Output	BLEU
Reference	All the witnesses, both congressmen and industry experts, expressed support for the bill in order to prevent profiteers from cashing in on airline profits at the expense of safe, cost-effective service.	
lang	*at the expense safe, cost-effective service of cashing in on profits airline from profiteers to prevent in order witnesses, all the, both industry experts and congressmen expressed support for the bill.*	0.3158
gf	all the witnesses, both congressmen and industry experts, expressed *for the bill support order in* to prevent profiteers from cashing in *at of cost-effective service, safe the expense on airline profits.*	0.5574
+head	*in order to prevent profiteers from cashing in at of cost-effective service, safe the expense on airline profits. all the witnesses, both congressmen and industry experts, expressed for the bill support*	0.6134
+parent	all the witnesses, both congressmen and industry experts, expressed *for the bill support* in order to prevent profiteers from cashing in *at of cost-effective service, safe the expense on airline profits.*	0.6134
+feat	all the witnesses, both congressmen and industry experts, expressed support for the bill in order to prevent profiteers from cashing in *at the expense of cost-effective, safe service on airline profits.*	0.7846
+lex	all the witnesses, both congressmen and industry experts, expressed support for the bill in order to prevent profiteers from cashing in on airline profits at the expense of *cost-effective service, safe.*	0.8915
all	all the witnesses, both congressmen and industry experts, expressed support for the bill in order to prevent profiteers from cashing in on airline profits at the expense of *cost-effective, safe* service.	0.8915
all+lang	all the witnesses, both congressmen and industry experts, expressed support for the bill in order to prevent profiteers from cashing in on airline profits at the expense of safe, cost-effective service.	1.0000

The results in Table 5 follow pretty much the same pattern as in Table 4: all the DN-gram models capturing some type of dependency relations outperform the simple language model, the combination of individual DN-gram models in turn makes further substantial improvements, and again, addition of the language model to the complex feature model gives the best performance with respect to generation accuracy. More importantly, the results show that the DN-gram models do not overfit the training data acquired from gold treebank trees. Even though the dependency trees automatically produced by the MSTParser are noisy (McDonald *et al.* (2005) reported unlabelled accuracy of 90.9 per cent discounting punctuation marks testing on Penn-II WSJ23), there is only a slight decrease (approximately 2 points BLEU and SSA absolute) in the realisation quality compared to dependencies derived from gold trees as generation input.

Another obvious observation is that all the evaluation scores for the CoNLL-style dependency structures (including parser output) are substantially higher than those for the LFG f-structures. On the one hand, the results indicate that the

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

dependency-based n-gram generation models generalise well to different realisation inputs, on the other hand they also reaffirm the statement by many researchers that generation results are heavily dependent on how specific the generation inputs are. We note that CoNLL dependency structures and LFG f-structures exhibit different levels of linguistic abstraction. The CoNLL dependencies tend to be more 'surfacy' and fine-grained than the more abstract LFG f-structures:

- Labels of the CoNLL dependencies are more fine-grained than f-structure grammatical functions, and manifest some surface syntactic features, for example CoNLL-style modifiers are classified into modifiers of adjective or adverbial (AMOD) heads, modifiers of nominal (NMOD) heads and modifiers of prepositions (PMOD).
- Part-of-speech tags are fully specified in the CoNLL dependency representations and used as a syntactic feature by the *feat* model, which has a strong positive effect on the realisation quality. By contrast, the LFG models do not use POS information.
- Nodes of the CoNLL dependency representations are simply the word forms occurring in surface sentences rather than lemmas as encoded in the predicates of the LFG f-structures.
- Function words are explicitly represented as nodes in the CoNLL dependency representations, and only indirectly as atomic-valued features in the LFG f-structures.

It might be argued that the CoNLL dependency representations do not provide 'semantic' specifications, and therefore they are not fully theoretically adequate representations for sentence realisation. However, as outputs of many off-the-shelf and state-of-the-art dependency parsers, the CoNLL-like dependency structures are certainly important representations available to practical NLG systems (including transfer-based machine translation), they have become a de-facto standard in many applications.

5 Comparison with previous work

As has been discussed in the previous section and pointed out by many researchers working in wide-coverage sentence realisation (Langkilde 2002; White and Rajkumar 2009), it is notoriously difficult to compare sentence realisers, as generation quality can vary widely due to the diverse application-specific inputs. The purpose of this section is therefore not to provide a strict comparison, but to rather draw an overall picture that relates our general-purpose sentence realiser to the state-of-the-art in surface realisation. For this reason, we presented systems (trained and) tested on the same data sets (although with a wide variety of input specifications) and evaluated using the same automatic evaluation metrics. In addition to accuracy metrics (exact match, BLEU and SSA), we also include two metrics to evaluate generation coverage: *coverage* is the percentage of input representations for which the surface realiser can produce a (complete or partial) string, *complete* is the percentage of grammatically complete sentences realised. Table 7 summarises the results on Penn-II WSJ23 as reported in the NLG literature.

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

Table 7. *Cross-system comparison of results for English WSJ section 23*

System	Coverage	Complete	ExMatch	BLEU	SSA
Callaway (2003)	98.7%		49.0%		0.8884
Langkilde (2002)[a]	82.7%		28.2%	0.757	0.696
Nakanishi *et al.* (2005)[b]	90.75%			0.7733	
Cahill and van Genabith (2006)	98.05%	89.49%		0.6651	0.6808
Hogan *et al.* (2007)	99.96%			0.6882	0.7092
Rajkumar *et al.* (2009)	94.8%	85.04%	33.74%	0.8173	
White and Rajkumar (2009)	97.06%	83.88%	40.45%	0.8506	
Guo *et al.* (2008)	100%	100%	19.83%	0.7440	0.7534
This article LFG	100%	100%	31.54%	0.8065	0.7871
This article CoNLL	100%	100%	47.76%	0.8820	0.8596

[a] The results are for the 'permute, no dir'-type experiment in Langkilde (2002), where the inputs are most comparable to our f-structures with regard to the level of specification.
[b] The results are for sentences with a length limitation of 20 words.

The surface realisers listed in Table 7 are broadly classified into four groups according to the generation methodology adopted by them. The system present in Callaway (2003) is a successor to the symbolic surface realiser FUF/SURGE, which is fully developed manually. HALOGEN (Langkilde 2002) is a two-stage hybrid realiser composed of a grammar-based symbolic generator and a statistical ranker. The five systems listed in the middle part of the table are statistical realisers using bi-directional, wide-coverage, probabilistic grammars automatically derived from the Penn-II treebank, couched in the frameworks of HPSG (Nakanishi *et al.* 2005), LFG (Cahill and van Genabith 2006; Hogan *et al.* 2007) and CCG (Rajkumar, White and Espinosa 2009; White and Rajkumar 2009). Guo *et al.* (2008) is a precursor of the research presented in this paper generating strings directly from the LFG f-structure representations using the dependency n-gram models (rather than grammar rules). The early work differs in three major respects from the work presented in this paper: it uses simple linear interpolation (rather than the log-linear combinations presented here), it does not factor in a word-based language model and it uses the LFG inputs only (rather than the CoNLL representations).

With respect to generation coverage (both partial and complete), our dependency n-gram-based realisers are the most robust. As they can always make a prediction with respect to the linear precedence of input dependencies, they guarantee a complete realisation for each input. By contrast, for the grammar-based surface realisers, limited sets of grammar rules, either derived from treebanks or developed manually, sometimes fail to cover the generation input and more often produce partial realisations or even no realisation output at all.

As for accuracy, it is surprising (or not) that the best performing system is a purely symbolic realiser. However, Callaway (2003) reports that it took several months to optimise the system tuning the available SURGE grammar to the Penn-II treebank data, and this on top of the total time required for original grammar development. Because of the considerable manual effort involved in development, surface realisers

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

based on handcrafted rules are difficult to adapt to different languages, domains and applications. The hybrid realiser HALOGEN achieves respectable accuracy, but also contains a substantial component based on symbolic knowledge that is time-consuming and non-trivial to develop.

There are a number of systems that automatically acquire reversible grammars from treebanks, and hence require less development effort than approaches that use handcrafted wide-coverage grammatical resources. In this particular respect, treebank-grammar-based generators are as efficient as the direct dependency n-gram-based linearisation models.

The most direct comparison in Table 7 is probably between our dependency n-gram-based systems and the treebank-grammar-based systems presented in Cahill and van Genabith (2006) and Hogan *et al.* (2007), because all these generators take as inputs the LFG f-structures automatically generated from the Penn-II treebank. There is an obvious gap in the accuracy scores between our DN-gram model and the two grammar-based realisers, which is, to a large extent, due to the low-generation coverage of the two treebank LFG grammar-based realisers. Though Cahill and van Genabith (2006) report that 98.05 per cent inputs receive a realisation, the number drops to 89.49 per cent when counting only complete realisations that fully cover the input representations, which means that nearly 10 per cent of the outputs of their realiser are in fact partial strings. The generation models described in Cahill and van Genabith (2006) and Hogan *et al.* (2007) are both conditional PCFG models in the sense that they use (LFG f-structure annotated) the CFG rules extracted from the Penn-II treebank with probabilities and parameters heavily conditioned on the input f-structure features. In order to partly compensate for the low coverage, we recast their conditional generation models as (properly) the generative PCFG models by ignoring global (and as yet unrealised) conditioning f-structure features from the generation input. The generative PCFG model substantially increases complete coverage to 96.44 per cent and as a natural consequence, the BLEU score is boosted to 0.7458 and the SSA score to 0.7456. The accuracy scores achieved by the generative PCFG model are roughly comparable to those obtained by the early DN-gram model described in Guo *et al.* (2008), but these are still substantially behind the more advanced models presented in this paper.

The 0.7733 BLEU score reported in Nakanishi *et al.* (2005) is relatively high, but applies to sentences of up to 20 words only in the WSJ section 23 test set, and the BLEU score for 500 sentences without a length limitation randomly selected from the WSJ section 22 development set drops to 0.705 with a lower coverage of 83.6 per cent.

The two CCG realisers employ a lexico-grammar extracted from the enhanced CCG bank, a hypertagging model incorporating named entity (NE) class features, and a combined trigram language model over words, NE classes, part-of-speech tags and supertags/categories to generate and score realisation candidates. Finally, a perceptron model integrated with a comprehensive feature set is used for re-ranking the n-best realisations produced by the trigram language models. The sophisticated CCG realisation models yield very attractive accuracy scores, though with relatively lower complete coverage. Although these models are designed specific to CCG, some

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

of the techniques, such as better handling of named entities as presented in Rajkumar *et al.* (2009) and perceptron re-ranking described in White and Rajkumar (2009), are compatible with the dependency-based approach proposed in this paper and we will pursue those in the further research as well.

Another important criterion for assessing a practical NLG system is generation time – the time required to generate one sentence on average. In theory, our DN-gram-based realisation model does not exhibit time complexity superior to other grammar-based realisers. For linearising n dependencies in a local dependency tree, the Viterbi algorithm finds the most probable linearisation among all permutations in time $O(n^2 2^n)$. Even though this is an exponential time model, as shown in Table 3, the average number of GFs/dependencies in a local f-structure (FstrLen n) for the treebank data is only about 3 for both English and Chinese. In practice this translates into very reasonable execution times. All experiments reported in this paper are run on a server with a 32-bit quad-core Intel Xeon 2.33-GHz processor and 8-GB memory. It takes around 35 minutes to generate all 2,416 sentences of the WSJ section 23 (an average of 0.87 seconds per sentence) and approximately 5 minutes to generate 1,900 sentences of the CTB6 test data (an average of 0.16 seconds per sentence). The difference in generation times between English and Chinese is mainly due to the fact that the maximum branching factor of the English f-structures is higher than that of Chinese. There are two English f-structures in the test data with sixteen dependencies at a local f-structure, which take more than two-thirds of the total generation time (26 minutes) to realise. Local f-structures like these tend to be coordinations or enumerations. In fact, one of these extreme cases is a list of American state names and the other a list of company names, where the order between the names does not have a substantial effect on the grammaticality of the final realisation. This suggests that a better strategy for treating coordinations should further improve the efficiency of our DN-gram-based surface realiser: simply removing the two outliers reduces the average realisation times to 0.22 seconds per sentence for the English data. Neither Cahill and van Genabith (2006) nor Hogan *et al.* (2007) report generation times in their experiments. Our reimplementation of their conditional PCFG generation model realises sentences for WSJ23 in 44 minutes (1.09 seconds per sentence), which is on the same level as our DN-gram model. However, enlarging the generation coverage by licensing more grammar rules in the generative PCFG model causes a combinatorial explosion and drastically increases the total generation time to almost 20 hours, viz. average 29.80 seconds per sentence. Actually, we find that this is a common phenomenon in wide-coverage chart-style sentence realisers. Nakanishi *et al.* (2005) report an average run time of 27.9 seconds per sentence for realising the 500 sentences selected from the WSJ section 22, running on the AMD Opteron servers with a 2.0-GHz CPU and 12-GB memory. The 'permute, no dir' experiment in Langkilde (2002) results in average generation time of 29.8 seconds per sentence. Given a 15-second per sentence time limit, White and Rajkumar (2009) report average generation time of 3.6 seconds on the development section 00. Callaway (2003) reports an average 3.48 seconds per sentence testing on WSJ23 for his symbolic realiser.

On the whole, our general-purpose direct DN-gram surface realiser ranks well within the best-performing systems, with fast generation speeds, complete coverage

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

and very competitive BLEU, SSA and exact match scores. The DN-gram-based model is simple, as it does not involve a grammar- and chart-based intermediate generation step (as many other approaches) but learns to linearise dependencies directly. Furthermore, it is generic, rather than tied to particular input representations, as many semantic or abstract syntactic input representations can be construed as dependencies, and a DN-gram model can be trained for them. In this paper we exemplified this using both LFG- and CoNLL-style dependency representations. To the best of our knowledge, this is the first-generation model for the CoNLL-style dependencies, complementing and extending the extensive work in dependency parsing (McDonald *et al.* 2005; Nivre 2006) based on these representations to generation. Our realiser strikes a good balance between accuracy, coverage, robustness and efficiency (both for development and generation), and we show that the method adapts well to different languages (English and Chinese).

6 Conclusions and further work

Sentence generators are usually built for specific semantic or abstract syntactic input representations. In this paper we describe a general-purpose, wide-coverage probabilistic sentence generator for realising sentences from labelled bi-lexical dependencies. The approach is generic, as many input representations can be captured in terms of dependencies. The sentence realiser maps directly from input specifications to surface strings by a number of increasingly sophisticated dependency n-gram models incorporating various structural and lexical features derived from the dependency representations. The direct linearisation method is simple compared to realisation models that involve building surface syntactic trees based on either treebank-based or handcrafted grammars as an intermediate step in generation. The DN-gram-based generation models have been tested on two different languages – English and Chinese, as well as on two dependency representations – the more abstract and the theoretically motivated LFG f-structures and the more practically oriented CoNLL-style dependencies. The dependency-based n-gram models are easy to train and execute quickly in practical applications. The models are robust with complete generation coverage and competitive generation accuracy measured in BLEU, SSA and exact match scores. Our best DN-gram model achieves 0.8065 BLEU, 0.7871 SSA and 31.54 per cent exact match on the English Penn-II data using the LFG f-structure input, 0.7713 BLEU, 0.7484 SSA and 32.89 per cent exact match for the Chinese CTB data using the LFG f-structure input, and 0.8820 BLEU, 0.8596 SSA and 47.76 per cent exact match for the CoNLL dependency input on the English Penn-II data, all with 100 per cent coverage and average generation times of about 0.87 to 0.16 seconds per sentence for our English and Chinese data, respectively. The experimental results show that the realisation method is general, flexible and easily portable to new languages, and hence to new domains and applications. To the best of our knowledge, our research provides the first surface realisation model for the CoNLL-style dependencies, complementing and extending the extensive work in dependency parsing (McDonald *et al.* 2005; Nivre 2006) based on these representations to generation. We believe that high-accuracy, real-time,

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

wide-coverage realisers will contribute to real-world applications, including transfer-based statistical machine translation with dependency-based transfer representations.

At the current stage, our general-purpose sentence realiser does not employ any special tactics for handling coordination structures, which seems to be one of the major reasons limiting SSA scores. We believe that a finer discrimination between different types of coordinations should improve generation quality. From a more practical point of view, packing equivalent conjuncts (with equal linear precedence) into a single class at a local dependency structure and (simply) unfolding them into one of the acceptable realisations or producing n-best alternatives as output would certainly benefit the efficiency of the sentence generator.

The DN-gram-based surface realiser has been demonstrated to generalise well in practice with respect to types of languages and input representations. Nevertheless, the DN-gram models are more or less limited to the specific features selected and integrated in the models, as well as to the ways adopted to combine models as described in this paper. Bilmes and Kirchhoff (2003) present an interesting theoretical framework, factored language models (FLM) with generalised parallel backoff, where a word is decomposed into a vector of various linguistic factors and a probabilistic model over the set of factors is induced. Considering the flexibility in unifying heterogeneous types of features, it is a promising direction for future research to apply FLM to linearising dependencies for sentence realisation. We hope that a wider variety of features extracted from input dependency structures can be integrated in a factored n-gram model, and an optimal combination of the features can be automatically learned using generalised backoff.

Acknowledgments

We are grateful to the anonymous reviewers for carefully reviewing this paper and providing very insightful comments. Part of the research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Dublin City University.

References

Bangalore, S., and Rambow, O. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th International Conference on Computational Linguistics*, pp. 42–48. Saarbrücken, Germany.

Bateman, J. A. 1997. Enabling technology for multilingual natural language generation: the KPML development environment. *Journal of Natural Language Engineering* **3**(1): 15–55. (Cambridge University Press)

Belz, A. 2007. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Journal of Natural Language Engineering* **1**(1): 1–26. (Cambridge University Press)

Bilmes, J. A. and Kirchhoff, K. 2003. Factored language models and generalized parallel backoff. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pp. 4–6. Edmonton, Canada.

Cahill, A., Burke, M., O'Donovan, R., van Genabith, J., and Way, A. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

LFG approximations. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pp. 320–327. Barcelona, Spain.

Cahill, A., Forst, M., and Rohrer, C. 2007. Stochastic realisation ranking for a free word order language. In *Proceedings of the 11th European Workshop on Natural Language Generation*, pp. 17–24. Schloss Dagstuhl, Germany.

Cahill, A., and van Genabith, J. 2006. Robust PCFG-based generation using automatically acquired LFG approximations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pp. 1033–1040. Sydney, Australia.

Callaway, C. B. 2003. Evaluating coverage for large symbolic NLG grammars. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 811–817. Acapulco, Mexico.

Carroll, J., Copestake, A., Flickinger, D., and Poznanski, V. 1999. An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation*, pp. 86–95. Toulouse, France.

Corston-Oliver, S., Gamon, M., Ringger, E., and Moore, R. 2002. An overview of Amalgam: a machine-learned generation module. In *Proceedings of the 2nd International Natural Language Generation Conference*, pp. 33–40. Harriman, NY.

Crouch, D., Dalrymple, M., Kaplan, R., King, T., Maxwell, J., and Newman, P. 2007. *XLE Documentation*. California: Palo Alto Research Center.

Dalrymple, M., Kaplan, R., Maxwell, J., and Zaenen, A. 1995. *Formal Issues in Lexical-Functional Grammar*, CSLI Lecture Notes No. 47. Standford, CA: CSLI Publications.

DeVault, D., Traum, D., and Artstein, R. 2008. Practical grammar-based NLG from examples. In *Proceedings of the 5th International Natural Language Generation Conference*, pp. 77–85. Salt Fork, OH.

Elhadad, M. 1993. FUF: the universal unifier user manual version 5.2. Technical Report, Department of Computer Science, Ben Gurion University of the Negev, Israel. http://www.cs.bgu.ac.il/surge/index.html.

Elhadad, M., and Robin, J. 1996. An overview of SURGE: a reusable comprehensive syntactic realization component. Technical Report 96-03, Mathematics and Computer Science Department, Ben Gurion University of the Negev, Israel.

Filippova, K., and Strube, M. 2007. Generating constituent order in German clauses. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 320–327. Prague, Czech Republic.

Filippova, K., and Strube, M. 2009. Tree linearization in English: improving language model-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 225–228. Boulder, CO.

Gamon, M., Ringger, E., Zhang, Z., Moore, R., and Corston-Oliver, S. 2002. Extraposition: a case study in German sentence realization. In *Proceedings of the 19th International Conference on Computational Linguistics*, pp. 1–7. Taipei, Taiwan.

Goodman, J. T. 2001. A bit of progress in language modeling. *Computer Speech and Language* **15**(4): 403–434.

Guo, Y., van Genabith, J., and Wang, H. 2007. Treebank-based Acquisition of LFG Resources for Chinese. In *Proceedings of LFG07 Conference*, pp. 214–232. Stanford, CA.

Guo, Y., van Genabith, J., and Wang, H. 2008. Dependency-based N-gram models for general purpose sentence realisation. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pp. 297–304. Manchester, UK.

Guo, Y., Wang, H., and van Genabith, J. 2010. A linguistically inspired statistical model for Chinese punctuation generation. *ACM Transactions on Asian Language Information Processing* **9**(2): 1–27.

Halliday, M. A. K. 1978. *Language as Social Semiotic: The Social Interpretation of Language and Meaning*. Maryland: University Park Press.

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

Hogan, D., Cafferkey, C., Cahill, A., and van Genabith, J. 2007. Exploiting multi-word units in history-based probabilistic generation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning*, pp. 267–276. Prague, Czech Republic.

Johansson, R., and Nugues, P. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of the 16th Nordic Conference of Computational Linguistics*, pp. 105–112. Tartu, Estonia.

Kaplan, R., and Bresnan, J. 1982. Lexical functional grammar: a formal system for grammatical representation. *The Mental Representation of Grammatical Relations*, pp. 173–282. Cambridge, MA: MIT Press.

Kaplan, R., and Wedekind, J. 2000. LFG generation produces context-free languages. In *Proceedings of the 18th International Conference on Computational Linguistics*, pp. 425–431. Saarbrücken, Germany.

Kay, M. 1979. Functional grammar. In *Proceedings of the 5th Annual Meeting of the Berkeley Linguistics Society*. Berkeley, CA.

Klakow, D. 1998. Log-linear interpolation of language models. In *Proceedings of the 5th International Conference on Spoken Language Processing*, vol. 5, pp. 1695–1699. Sydney, Australia.

Langkilde, I. 2000. Forest-based statistical sentence generation. In *Proceedings of 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 170–177. Seattle, WA.

Langkilde, I. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the 2nd International Conference on Natural Language Generation*, pp. 17–24. Harriman, NY.

Langkilde, I., and Knight, K. 1998. The practical value of N-grams in derivation. In *Proceedings of the 9th International Workshop on Natural Language Generation*, pp. 248–255. New Brunswick, NJ.

Lavoie, B., and Rambow, O. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pp. 265–268. Washington, DC.

Marciniak, T., and Strube, M. 2004. Classification-based generation using TAG. In *Proceedings of the 3rd International Conference on Natural Language Generation*, pp. 100–109. Brockenhurst, UK.

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

McDonald, R., Crammer, K., and Pereira, F. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pp. 91–98. Ann Arbor, MI.

Mel'čuk, I. A. 1988. *Dependency Syntax: Theory and Practice*. New York, NY: SUNY Press.

Nakanishi, H., Nakanishi, Y., and Tsujii, J. 2005. Probabilistic models for disambiguation of an HPSG-based chart generator. In *Proceedings of the 9th International Workshop on Parsing Technology*, pp. 93–102. Vancouver, British Columbia, Canada.

Nicolov, N., and Mellish, C. 2000. PROTECTOR: efficient generation with lexicalized grammars. In *Recent Advances in Natural Language Processing II*, pp. 221–243. Amsterdam, The Netherlands: John Benjamins.

Nivre, J. 2006. *Inductive Dependency Parsing*. New York, NY: Springer.

Och, F. J. 2003 Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pp. 160–167. Sapporo, Japan.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318. Philadelphia, PA.

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press

Rajkumar, R., White, M., and Espinosa, D. 2009. Exploiting named entity classes in CCG surface realization. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 161–164. Boulder, CO.

Ratnaparkhi, A. 2000. Trainable methods for natural language generation. In *Proceedings of the 6th Applied Natural Language Processing Conference and 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 194–201. Seattle, WA.

Reiter, E., and Dale, R. 1997. Building applied natural language generation systems. *Journal of Natural Language Engineering* **3**: 57–87. (Cambridge University Press)

Ringger, E., Gamon, M., Smets, M. E., Corston-Oliver, S., and Moore, R. C. 2003. Linguistically informed statistical models of constituent structure for ordering in sentence realization. Technical Report MSR-TR-2003-54, Microsoft Research, Redmond, WA.

Stolcke, A. 2002. SRILM-An extensible language modeling toolkit. In *Proceedings of the 7th International Conference of Spoken Language Processing*, pp. 901–904. Denver, CO.

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. 2008. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the 12th Conference on Computational Natural Language Learning*, pp. 159–177. Manchester, UK.

Uchimoto, K., Murata, M., Ma, Q., Sekine, S., and Isahara, H. 2000. Word order acquisition from corpora. In *Proceedings of the 18th International Conference on Computational Linguistics*, pp. 871–877. Saarbrücken, Germany.

Velldal, E., and Oepen, S. 2005. Maximum entropy models for realization ranking. In *Proceedings of the 10th Machine Translation Summit*, pp. 109–116. Phuket, Thailand.

White, M. 2004. Reining in CCG chart realization. In *Proceedings of the 3rd International Natural Language Generation Conference*, pp. 182–191. Brockenhurst, UK.

White, M., and Rajkumar, R. 2009. Perceptron reranking for CCG realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 410–419. Singapore.

White, M., Rajkumar, R., and Martin, S. 2007. Towards broad coverage surface realization with CCG. In *Proceedings of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation*, pp. 22–30. Copenhagen, Denmark.

Xue, N., Xia, F., Chiou, Fu dong, and Palmer, M, 2005. The penn Chinese TreeBank: phrase structure annotation of a large corpus. *Natural Language Engineering* **11**(2): 207–238. (Cambridge University Press)

Zhong, H., and Stent, A. J. 2005. Building surface realizers automatically from corpora. In *Proceedings of the Corpus Linguistics 2005 Workshop on Using Corpora for Natural Language Generation*, pp. 49–54. Birmingham, UK.

https://doi.org/10.1017/S1351324910000288 Published online by Cambridge University Press