Syntax and Data-to-Text Generation

Claire Gardent

Syntax and Data-to-Text Generation

Claire Gardent

CNRS/LORIA
Nancy, France
claire.gardent@loria.fr

Abstract. With the development of the web of data, recent statistical, data-to-text generation approaches have focused on mapping data (e.g., database records or knowledge-base (KB) triples) to natural language. In contrast to previous grammar-based approaches, this more recent work systematically eschews syntax and learns a direct mapping between meaning representations and natural language. By contrast, I argue that an explicit model of syntax can help support NLG in several ways. Based on case studies drawn from KB-to-text generation, I show that syntax can be used to support supervised training with little training data; to ensure domain portability; and to improve statistical hypertagging.

Keywords: Computational Grammars, Natural Language Generation, Statistical Natural Language Processing, Hybrid Symbolic/Statistical Approaches

1 Introduction

Given some non-linguistic input, the task of data-to-text generation consists in producing a text verbalisating that input. Data-to-text generation has been used, e.g., to summarise medical data [31], to generate weather reports from numerical data [32] and to automatically produce personalised letters [11].

Earlier statistical work on data-to-text generation has mainly focused on inducing large probabilistic grammars from treebanks and on using these grammars to generate from meaning representations derived from those same treebanks. Thus, [9] induces a Probabilistic Lexical Functional Grammar (LFG) from the PTB and uses it to generate from f(unctional)-structures automatically derived from that treebank. [3] uses a large scale Tree Adjoining Grammar (TAG, [34]) and a tree model trained on the derivation trees of 1 million words of the Wall Street Journal to map dependency trees to sentences. And [38] induces a probabilistic Combinatory Categorial Grammar (CCG, [33]) from the CCGBank [21] which is then used to generate from hybrid logic dependency semantics [2].

With the development of the web of data however, interest has recently shifted to data-to-text generators which can generate from less linguistic, more data oriented, meaning representations. While logical formulae and

dependency trees may provide generic meaning representations for natural language, they typically fail to support a straightforward mapping between data and natural language (NL) expressions. This is because both the signature of the meaning representation language and the alignment between meaning and basic grammar units are specified independently of the application data. Typically, predicate names are simply lemmas (each word will be represented by a meaning representation including its lemma as a predicate symbol) and the alignment between meaning and string is determined by syntax. When generating from e.g., knowledge or database data, these assumptions generally fail to hold. That is, lemmas must be disambiguated and mapped to application-specific predicate symbols while the alignment between meaning representation sub-units and NL expressions is often at odd with grammar syntax.

To address these issues, recent statistical, data-to-text approaches have therefore focused on mapping e.g., database records or knowledge-base (KB) triples to natural language. In particular, data-to-text generators [1, 10, 39, 24, 23] were trained and developed on datasets from various domains including the air travel domain [13], weather forecasts [26, 5] and sportscasting [10]. In contrast to the previous, grammar-based approaches, this more recent work systematically eschews syntax. Instead, the dominant approach consists in learning a direct mapping between meaning representations and natural language.

In this paper, we take a middleroad between these two approaches. We focus on generating from "real" data i.e., knowledge base data, but we argue that an explicit model of syntax is valuable in several ways. More specifically, we argue that syntax:

- *can help compensate for the lack of large quantities of training data.* Using an international benchmark consisting of only 207 training instances, we show that inducing a linguistically principled, non probabilistic grammar from this data, allows for the development of a data-to-text generator which shows good coverage while preserving output quality. When compared with the other two participating systems, the approach performs comparably with a rule-based, manually developed system and markedly outperforms an existing statistical generator.
- *can help ensure genericity.* Focusing on the task of verbalising user queries on knowledge bases, we show that a small hand-crafted grammar, combined with an automatically constructed lexicon, permits verbalising queries independent of which domain the queried KB bears on.
- *can help improve the performance of a statistical, hypertagging module* designed to reduce the initial search space of the generator. In particular, we show that the high level linguistic abstractions captured by the grammar permits developing a hypertagging module which improves the generator speed, supports sentence segmentation and preserves output quality.

The paper is structured as follows. In section 2, we start by introducing the grammar framework which we use to support data-to-text generation namely, Feature-Based Lexicalised Tree Adjoining Grammar (FB-LTAG). We then explain the generation algorithm which permits gen-

erating sentences given some input data and an FB-LTAG. Sections 3, 4 and 5 illustrate how an explicit model of syntax can help improve generation. Section 6 concludes with pointers for further research.

2 Feature-Based Lexicalised Tree Adjoining Grammar

In this section, we start by defining the grammar formalism (Section 2.1) and the lexicon (Section 2.2) we use to mediate between data and natural language. We then describe the generation algorithm which exploits these lexicon and grammar to map data into text (Section 2.3).

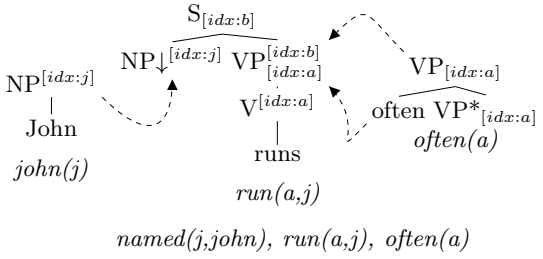

Fig. 1. Derivation and Semantics for "John often runs"

2.1 Grammar

Following [17], we use a Feature-Based Lexicalised Tree Adjoining Grammar (FB-LTAG) augmented with a unification based semantics for generation. For a precise definition of FB-LTAG, we refer the reader to [36]. In essence, an FB-LTAG is a set of trees whose nodes are decorated with feature structures and which can be combined using either substitution or adjunction. Substitution of tree γ_1 at node n of the derived tree γ_2 rewrites n in γ_2 with γ_1. n must be a substitution node (marked with a downarrow). Adjunction of the tree β at node n of the derived tree γ_2 inserts β into γ_2 at n (n is spliced to "make room" for β). The adjoined tree must be an auxiliary tree that is a tree with a foot node (marked with a star) and such that the category of the foot and of the root node is the same.

In an FB-LTAG with unification semantics, each tree is furthermore associated with a semantics and shared variables between syntax and semantics ensure the correct mapping between syntactic and semantic arguments. As trees are combined, the semantics of the resulting derived tree is the union of their semantics modulo unification.

The semantic representation language used to represent meaning in the grammar is a flat semantics language [6, 12] which consists of a set of literals and can be used e.g., to specify first order logic formulae or RDF triples. For a precise definition of the syntax and semantics of that language, see [18].
Figure 1 shows an example toy FB-LTAG with unification semantics. The dotted arrows indicate possible tree combinations (substitution for *John*, adjunction for *often*). Thus given the grammar and the derivation shown, the semantics of *John often runs* is as shown namely, *named(j john), run(a,j), often(a).*

2.2 Lexicon

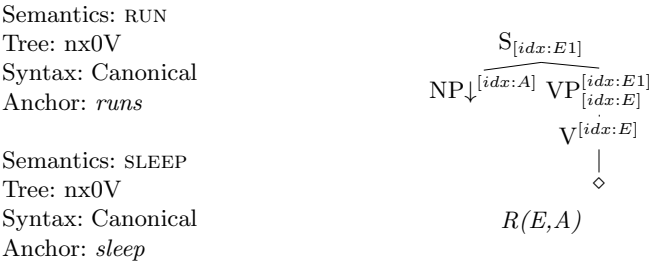

Fig. 2. FB-LTAG tree schema nx0V and two Lexical Entries associated with that tree schema.

The *Lexicon* permits abstracting over lexical and semantic information in an FB-LTAG tree and relating a single tree schema to several lexical items. For instance, the lexical entries shown on the left of Figure 2 relates the predicate symbols RUN and SLEEP to the TAG tree nx0V shown on the right. During generation, these relation predicate symbols will be used to instantiate the predicate variable R in the semantic schema $R(E, A)$ associated with that tree; and the anchor values (*runs/sleeps*) to anchor this tree i.e., to label the terminal node marked with the anchor sign ($\diamond$). That is, the $\diamond$ node will be labelled with the terminal *runs/sleeps*.

2.3 Surface Realisation

For surface realisation, we use the chart-based algorithm described in [19]. This algorithm proceeds in four main steps as follows.

- Lexical Selection. Retrieves from the grammar all grammar units whose semantics subsumes the input semantics. Fo instance, given the semantics *named(j john), run(a,j), often(a)*, lexical selection will return the three trees shown in Figure 1.

- Tree Combination. Substitution and adjunction are applied on the set of selected trees and on the resulting derived trees until no further combination is possible. For instance, the three trees selected in the previous lexical selection step will be combined to yield a complete phrase structure tree.
- Sentence Extraction. all syntactically complete trees which are rooted in S and associated with exactly the input semantics are retrieved. Their yields provide the set of generated (lemmatised) sentences e.g., *John run often* in our running example.
- Morphological Realisation. Lexical lookup and unification of the features associated with lemmas in the generated lemmatised sentences yield the final set of output sentences e.g., *John runs often*

3 Grammar as a Means to Compensate for the Lack of Training Data

The KBGen task[1] was introduced as a new shared task at Generation Challenges 2013 [4] to evaluate and compare systems that generate text from knowledge base data. Figure 3 shows an example input and output.

```
                                :TRIPLES (
    (|Release-Of-Calcium646| |object| |Particle-In-Motion64582|)
    (|Release-Of-Calcium646| |base| |Endoplasmic-Reticulum64603|)
    (|Gated-Channel64605|  |has-function||Release-Of-Calcium646|)
      (|Release-Of-Calcium646|  |agent| |Gated-Channel64605|))
                              :INSTANCE-TYPES
   (|Particle-In-Motion64582| |instance-of| |Particle-In-Motion|)
(|Endoplasmic-Reticulum64603| |instance-of| |Endoplasmic-Reticulum|)
       (|Gated-Channel64605| |instance-of| |Gated-Channel|)
    |Release-Of-Calcium646| |instance-of| |Release-Of-Calcium|))
                              :ROOT-TYPES (
         (|Release-Of-Calcium646| |instance-of| |Event|)
       (|Particle-In-Motion64582| |instance-of| |Entity|)
     (|Endoplasmic-Reticulum64603| |instance-of| |Entity|)
       (|Gated-Channel64605| |instance-of| |Entity|)))
```

The function of a gated channel is to release particles from the endoplasmic reticulum

Fig. 3. Example KBGEN Input and Reference Sentence

One characteristic of the KBGen shared task is that the size (207 input/output pairs) of the training data is relatively small which makes it difficult to learn efficient statistical approaches. In what follows, we show that, by inducing a linguistically principled grammar from the training data, we can develop a generator which performs well on the test data. While fully automatic, our approach produces results which are comparable to those obtained by a hand written, rule based system; and which markedly outperform a data-driven, generate-and-rank approach based on an automatically induced probabilistic grammar.

Grammar induction generally relies on large syntactically annotated corpora (treebank) and results in large grammars whose combinatorics are constrained by the probability estimates derived from the treebank. In contrast, we define a grammar induction algorithm which yields compact,

[1] `http://www.kbgen.org`

Algorithm 1 Grammar Induction Algorithm

Require: An input semantics (set of triples) ϕ with variables V_ϕ, reference sentence S and parse tree τ_S.

1: **Variable/String Alignment** Align each variable in V_ϕ with one or more tokens in S
2: **Variable Projection** Use the Variable/String alignment and a set of hand-written rules to project the input semantics variables onto non terminal nodes in the parse tree τ_S of the reference sentence.
3: **Extracting Trees** Extract NP and relational (prepositions, verbs, conjunctions, etc.) trees from τ_S using the variables labelling the nodes of the parse tree. NP trees are NP subtrees whose root node are labelled with an input variable ($v \in V_\phi$). Relational trees are subtrees containing all and only input variables that are related to each other by relations in ϕ (cf. [20] for a more precise definition).
4: **Associating Trees with Semantics**. Each subtree is assigned a set of input triples based on the input variables it is labeled with. An NP tree labeled with input variable v is associated with all input literal whose first argument is v. Each relational tree is associated with all literals whose argument variables are variables labelling this tree.
5: **Generalising from Trees to Tree Schemas** Isomorphic trees which differ only in their semantics and lexical content are converted to a single tree schema and several lexical entries capturing the multiple possible instantiations of that tree schema.
6: **Generalising from Bigger to Smaller Trees** Large Verb trees are used to derive smaller more general trees e.g., by deriving an intransitive verb tree from a transitive one; or by splitting a tree containing a PP into one tree without that PP and a PP tree.
7: **return** An FB-LTAG with unification semantics and a lexicon mapping semantic triples to FB-LTAG trees

linguistically principled, FB-LTAG grammars. The induction process is informed by the two main principles underlying the linguistic design of an FB-LTAG namely, the extended domain of locality and the semantic principle. The *extended domain of locality principle* requires that elementary TAG trees group together in a single structure a syntactic functor and its arguments while the *semantic principle* requires that each elementary tree captures a single semantic unit. Together these two principles ensure that TAG elementary trees capture basic semantic units and their dependencies.

Figure 1 gives a high level description of our grammar algorithm (see [20] for a more detailed description). In brief, the algorithm takes as input a set of (ϕ, S) pairs provided by the KBGen challenge where ϕ is a set of triples and S is a sentence verbalising ϕ. First, input variables in ϕ are aligned with word forms in S. Variables are then projected on non terminal nodes of S parse tree and used to constrain both tree extraction and the association of syntactic trees with semantics. Steps 5 and 6 of the algorithm generalise the extracted grammar by abstracting away from specific lexical and predicative information (Step 5) and by deriving smaller trees from extracted ones (Step 6).

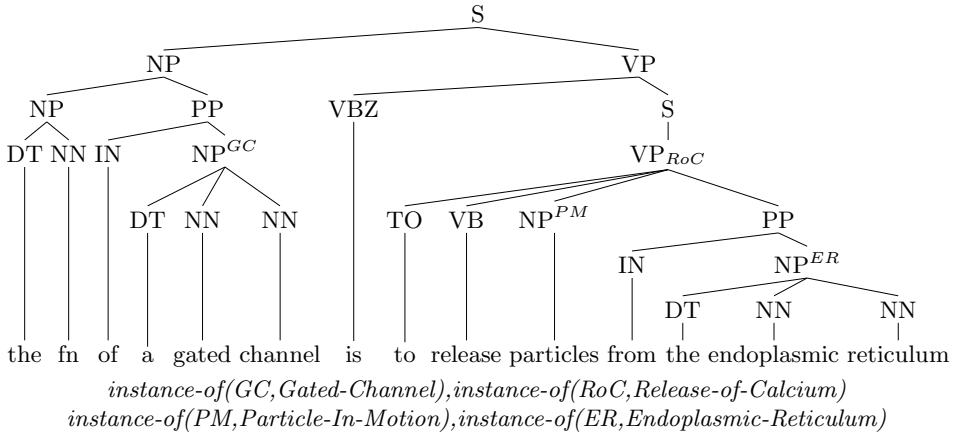

Fig. 4. Parse Tree with Projected Semantic Variables. Input Variables are first aligned with word forms (Step 1) and then projected onto parse tree nodes (Step 2).

Figure 4 shows a parse tree after variable projection and Figure 5 shows the grammar produced by Step 4. Generalisation is illustrated in Figure 6.

As illustrated by Figures 5 and 6, the grammars extracted by our grammar induction algorithm are FB-LTAGs which conform with the semantic and the extended domain locality principle. In [20], we show that inducing such grammars help compensate the small size of the training data. Tables 7 and 8 show the results obtained on the KBGen data and compare them with those of the other two participating systems namely, a symbolic system based on hand written rules (UDEL) and a statistical system (IMS) based on a probabilistic grammar extracted from the KBGen training data. As the results show, our approach provides an interesting middle way between the two approaches. On the one hand, it produces sentences that are comparable in quality with those generated by the symbolic UDEL system but it produces them in a fully automatic manner thereby eschewing the need for highly skilled and time consuming manual labour. On the other hand, it generates sentences of much higher quality than those output by the statistical system. In sum, by extracting a linguistically principled grammar, we achieved good quality output while eschewing the need for manual grammar writing.

4 Grammar as a Means to increase Domain Independence

We now turn to a second data-driven NLG application namely, the task of verbalising knowledge-base queries. Interfaces to knowledge bases which

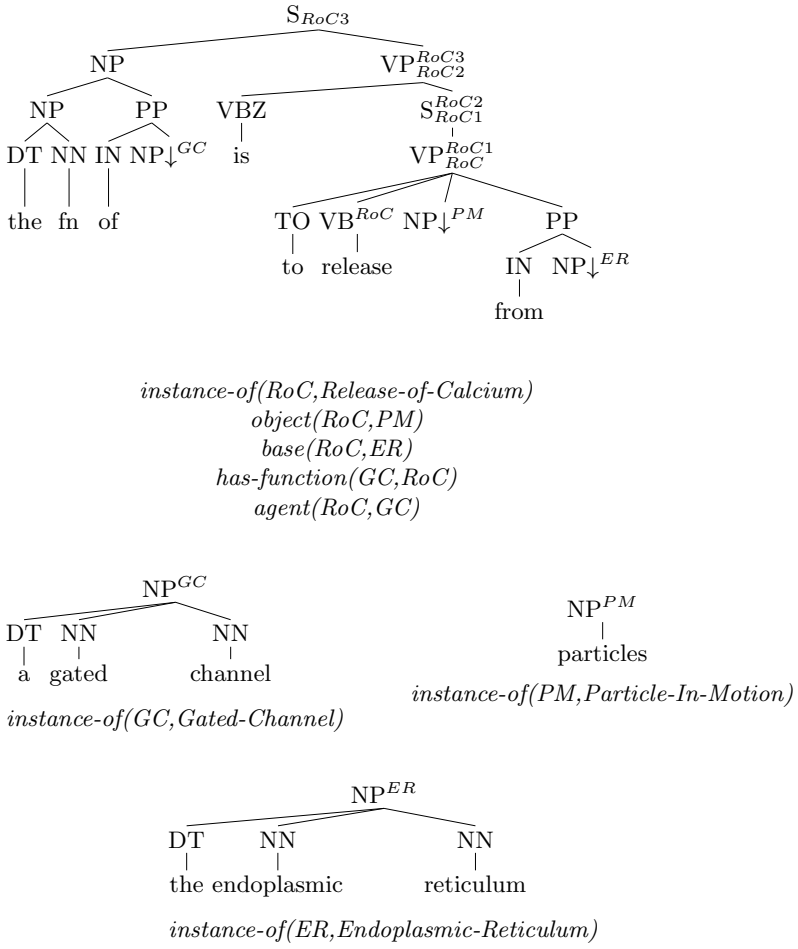

Fig. 5. Extracted Grammar for "*The function of a gated channel is to release particles from the endoplasmic reticulum*". Variable names have been abbreviated and the KBGen tuple notation converted to terms so as to fit the input format expected by our surface realiser.

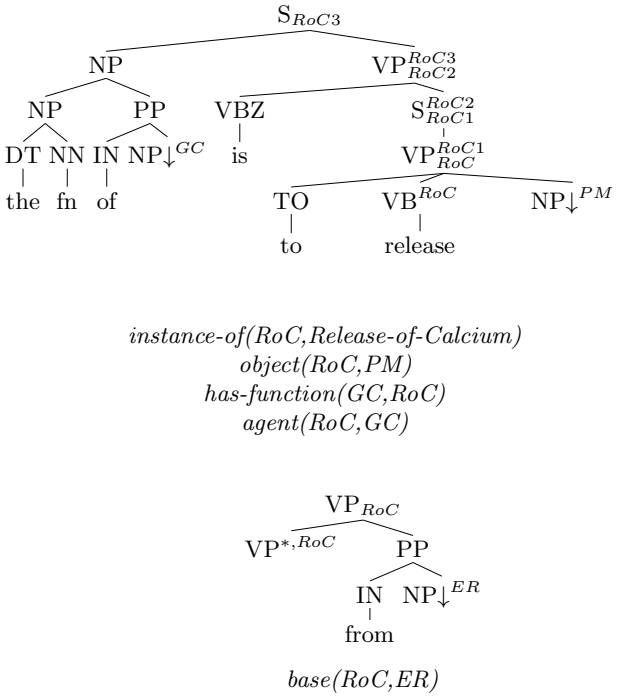

Fig. 6. Deriving Smaller from Larger Trees

System	All	Covered	Coverage	# Trees
IMS	0.12	0.12	100%	
UDEL	0.32	0.32	100%	
AutExp	0.29	0.29	100%	477

Fig. 7. BLEU scores and Grammar Size (Number of Elementary TAG trees

	Fluency		Grammaticality		Meaning Similarity	
System	Mean	Homogeneous Subsets	Mean	Homogeneous Subsets	Mean	Homogeneous Subsets
UDEL	4.36	A	4.48	A	3.69	A
AutExp	3.45	B	3.55	B	3.65	A
IMS	1.91	C	2.05	C	1.31	B

Fig. 8. Human Evaluation Results on a scale of 0 to 5. Homogeneous subsets are determined using Tukey's Post Hoc Test with $p < 0.05$

make use of Natural Language Generation have been shown to successfully assist the user by allowing her to formulate a query while knowing neither the formal query language nor the content of the KB being queried. In these interfaces, the user never sees the formal query. Instead, at each step in the query process, the generator verbalises all extensions of the current query which are consistent with this query and with the knowledge base. The user then chooses from among the set of generated NL queries, the query she intends.

One issue with such NLG based interfaces is that they should be domain independent. They should be usable for various KBs and various domains. Moreover, they should allow for an incremental processing of the user query. That is, they should support the user in incrementally refining her queries in a way that is consistent with the KB content.

While most previous work on generating from knowledge bases has assumed a restricted syntax and used either templates or a procedural framework (Definite Clause Grammars) to model the interaction between syntax, NL expressions and semantics, we developed an approach which uses a small hand-written FB-LTAG with unification semantics to support query verbalisation [30]. In essence, this approach consists in implementing a small hand-written FB-LTAG with unification semantics; in automatically constructing a lexicon which maps concept and relation names to trees in the FB-LTAG; and in adapting an Earley style parsing algorithm to support the incremental revision of a user queries.

In [30], we showed that this approach has several advantages.

First, because it does not rely on the existence of a training corpus, it is generic i.e., it results in a KB query system which can be used with different KBs on different domains. The key to developing a generic approach lies in combining a generic grammar which captures syntactic variations (canonical clauses, relative clauses, ellipses etc) with an automatically extracted lexicon which captures the lexicalisation of concepts and relations. While the grammar is hand written, the lexicon is automatically extracted from each ontology using the methodology described in [35]. When tested on a corpus of 200 ontologies, this approach was shown to be able to provide appropriate verbalisation templates for about 85% of the relation identifiers present in these ontologies. 12 000 relation identifiers were extracted from the 200 ontologies and 13 syntactic templates were found to be sufficient to verbalise these relation identifiers (see [35] for more details on this evaluation).

Thus, in general, the extracted lexicons permit covering about 85% of the ontological data. We further evaluated the coverage of our approach by running the generator on 40 queries generated from five distinct ontologies. The domains observed are cinema, wines, human abilities, disabilities, and assistive devices, e-commerce on the Web, and a fishery database for observations about an aquatic resource. The extracted lexicons contained 453 lexical entries in average and the coverage (proportion of formal queries for which the generator produced a NL query) was 87%. Fuller coverage could be obtained by manually adding lexical entries, or by developing new ways of inducing lexical entries from ontologies (c.f. e.g. [37]).

A second advantage of the grammar based approach is that because it uses a well-defined grammar framework rather than e.g., templates or a procedural framework, it allows for the use of existing generation techniques and algorithms. In particular, we showed that the Earley style generation algorithm proposed in [8] could straightforwardly be adapted to support the incremental generation of user queries.
Finally, using a fully blown approach to syntax rather than templates or programs allows for more syntactic variability and better control on syntactic and lexical interactions (e.g., by using features to ensure number agreement or control the use of elliptical constructions). In comparison, template based approach often generate one clause per relation[2]. Thus for instance, the template-based Quelo system [16] will generate (1a) while our grammar based approach supports the generation of arguably more fluent sentences such as (1b).

(1) a. I am looking for a car. Its make should be a Land Rover. The body style of the car should be an off-road car. The exterior color of the car should be beige.
b. I am looking for car whose make is a Land Rover, whose body style is an off-road car and whose exterior color is beige.

A human-based experiment indicate that the queries generated by our grammar based approach are perceived as more fluent than those produced by the Quelo template based approach (1.97 points[3] in average for the grammar based approach against 0.72 for the template based approach).

5 Grammar as a Means to Improve Statistical Disambiguisation

The generation algorithm described in Section 2.3 (i) explores the whole search space and (ii) is limited to generating a single sentence at a time. Generation from flat semantics is NP-complete however [7, 22] and existing algorithms for realization from a flat input semantics all have runtimes which, in the worst case, are exponential in the length of the input. Moreover, user queries can typically require the generation of several sentences. For instance, the query in (2) is better verbalised as (2a) than as (2b).

(2) a. CarMake(x) isMakeOf(x y) CrashCar(y) DemonstrationCar(y) hasCarBody(y z) OffRoad(z) soldBy(z w) CarDealer(w)
b. *I am looking for a car make which should be the make of a crash car, a demonstration car. The body style of the crash car should be an off road and it should be sold by a car dealer.*
c. *I am looking for a car make which should be the make of a crash car which is a demonstration car and whose body style should be an off road and should be sold by a car dealer.*

[2] This is modulo aggregation of relations. Thus two subject sharing relations may be realised in the same clause.
[3] Fluency was rated on a scale from 0 to 5.

In [29], we present a hypertagger which (i) restricts the search space output by the lexical selection step and (ii) segments the input into sentence size chunks. That is, our hypertagger not only restricts the initial search space thereby reducing timeouts and generation time, but it also permits a joint modelling of surface realisation and sentence segmentation. This is possible because, in contrast to approaches such as [27, 14, 24, 23] which directly map semantics to strings, we mediate this relation using a grammar which differentiates between sentence starting (e.g., nx0VVVpnx1 in Figure 1) and clause extending trees (e.g., relative clauses, sentence and VP coordination, PPs and elliptical clauses). Thus, a tagging sequence in effect determines sentence segmentation. For instance, given the query shown in (3a), if the hypertagger returns the sequence of tree tags shown in (3b), the output verbalisation will be (3d) because the tag Tnx0VVpnx1 indicates a sentence starting tree thereby forcing the segmentation of the input into two sentences.

(3) a. CarDealer(x) locatedIn(x,y) City(y) sell(x z) Car(z) runOn(z w) Diesel(w)
b. (Trees) Tnx betanx0VPpnx1 nx ANDWHnx0VVnx1 nx nx0VVpnx1 nx
c. (Synt.Classes) NP ParticipialOrGerund NP SubjRelAnd NP Canonical
d. *I am looking for a car dealer located in a city and who should sell a car. The car should run on a diesel.*

In practice however, we do not use tree names as hypertags because of data sparsity. The hypertagger is a model learned on a parallel corpus of NL and formal queries. Creating such a corpus is labour intensive and we only collected 145 training instances. A first experiment which predicts tree as labels yielded a tagging accuracy on complete inputs of 57.86 when considering the 10 best outputs. More importantly, the model often failed to predict a sequence which would allow for generation even when inputing to the tree combination phase the 10 best tree sequences predicted by the hypertagger.

The tags learned by our hypertagger are therefore not tree names but more general *syntactic classes* which capture the syntactic realisation of a semantic token independent of its lexical class. Thus, the input query in (3a) will, in fact, be tagged with the syntactic classes shown in (3c). Table 1 shows the tree names and the syntactic classes associated with each tree selected by the EQUIPPEDWITH relation while Table 2 shows example tree names for lexical classes with distinct subcategorisation patterns. As can be seen, while a tree name describes both the lexical and the syntactic pattern of a lexical item (e.g, nx0VVVnx1 describes a transitive verb with canonical subject and object NPs), syntactic classes capture syntactic generalisations which cut across all subcategorisation patterns (e.g.,the Canonical class is true of all realisations with canonical subject and object NPs). Since in our grammar, each tree is automatically associated by the grammar compilation process with its syntactic class, we first use the hypertagger to predict the syntactic class of an input literal. We then restrict the set of trees that were lexically selected by that literal to only those trees which have the syntactic class returned by the hypertagger. For instance, given the literal EQUIPPEDWITH, while lexical selection will return the set of trees shown in Table 1,

if the hypertagger predicts the SubjRelAnd class for this literal given the overall input, then the tree combination step of the generation algorithm will only consider the tree labeled with that syntactic class namely, the W0nx0VVVpnx1 tree.

Example	Tree Name	Syntactic Class
NP_0 should be equipped with NP_1	nx0VVVpnx1	Canonical
It_0 should be equipped with NP_1	PRO0VVVpnx1	SubjPro
and NP_0 should be equipped with NP_1	sCONJnx0VVVpnx1	Scoord
and it_0 should be equipped with NP_1	sCONJPRO0VVVpnx1	ScoordSubjPro
NP_0 which should be equipped with NP_1	W0nx0VVVpnx1	SubjRel
NP_0 (...) and which should be equipped with NP_1	ANDWHnx0VVVpnx1	SubjRelAnd
NP_0 (...), which should be equipped with NP_1	COMMAWHnx0VVVpnx1	SubjRelComma
NP_0 equipped with NP_1	betanx0VPpnx1	ParticipialOrGerund
NP_0 (...) and equipped with NP_1	betanx0ANDVPpnx1	ParticipialOrGerundAnd
NP_0 (...), equipped with NP_1	betanx0COMMAVPpnx1	ParticipialOrGerundComma
NP_1 with which NP_0 should be equipped	W1pnx1nx0VV	PObjRel
NP_0 (equipped with X) and with NP_1	betavx0ANDVVVpnx1	SubjEllipAnd
NP_0 (equipped with X), with NP_1	betavx0COMMAVVVpnx1	SubjEllipComma

Table 1. Verbalisations of the EQUIPPEDWITH relation captured by the lexicon and the grammar.

Hypertagging is viewed as a sequence labelling task in which the sequence of semantic input needs to be labelled with appropriate syntactic classes. The linear order of the semantic input is deterministically given by the linearisation process of the tree based conjunctive input (see [15] for more details).

We use a linear-chain Conditional Random Field (CRF, [25]) model to learn the mapping between observed input features and hidden syntactic classes. This probabilistic model defines the posterior probability of labels (syntactic classes) y=$\{y_1, \ldots, y_n\}$ given the sequence of input literals x=$\{x_1, \ldots, x_k\}$:

$$P(y \mid x) = \frac{1}{Z(x)} \prod_{t=1}^{T} exp \sum_{k=1}^{K} \theta_k \Phi_k(y_t, y_{t-1,x_t})$$

Z(x) is a normalisation factor and the parameters θ_k are weights for the feature functions Φ_k.

Given a set of candidate hypertags (syntactic classes) associated with each literal, the hypertagging task consists into finding the optimal hypertag sequence y^* for a given input semantics x:

$$y^* = argmax_{y^*} P(y^* \mid x)$$

whereby the most likely hypertag sequence is computed using the Viterbi algorithm. We used the Mallet toolkit [28] for parameter learning and inference.

NP_0 should generate NP_1	nx0VVnx1
NP_0 should run on NP_1	nx0VVpnx1
NP_0 should be equipped with NP_1	nx0VVVpnx1
NP_0 should be the equipment of NP_1	nx0VVDNpnx1
NP_0 should have access to NP_1	nx0VVNpnx1
NP_0 should be relevant to NP_1	nx0VVApnx1
NP_0 should be an N_1 product	nx0VVDNnx1

Table 2. Example of Canonical Trees for each Subcategorisation Class

We train the CRF on a corpus aligning formal queries with the syntactic classes present in the FB-LTAG grammar. The corpus contains 145 training instance with queries for 9 ontologies for different domains (car, cinema, wines, assistive devices and fishery).
All features are derived from the input semantics i.e., a sequence of interleaved relations and concepts. Since concepts have low lexical ambiguity (they mostly select NP trees), most of the features are associated with relations only and in the following, we write R_{i-1} to denote the relation which precedes relation R_i independently of how many concepts intervene between R_i and R_{i-1}. Features describe (i) the chaining relations between entities, (ii) the shape of relation names and correspondingly their lexicalisation properties i.e., the sequence of POS tags and indirectly the TAG tree that will be used to verbalise them, and (iii) global structural features pertaining to the overall shape of the input.
We evaluate the hypertagging module both in isolation and in interaction with the generator. The results show that hypertagging with syntactic classes[4]:

- improves hypertagging accuracy. Hypertagging with syntactic classes rather than tree names improves accuracy by up to 10.62 points for token accuracy; and by up to 20.77 points for input accuracy (token accuracy is the proportion of input literals correctly labelled while input accuracy is the proportion of correctly labelled input sequences).
- improves generation coverage by up to 17.25 points when compared with treename-based hypertagging (Generation coverage is the proportion of input for which generation yields an output).
- improves speed both with respect to both a generator using a treename-based hypertagger (-66ms in average per input) and a symbolic generator without hypertagging. The symbolic generator repeatedly times out yielding an average generation time of 17 minutes on 145 inputs.
- preserves output quality. When compared with both a grammar based and a template based generation system, the output of our hybrid statistical hypertagging/grammar-based generation system is

[4] For all results discussed, we assume a hypertagging module returning up to 20 best solutions.

consistently perceived by human raters as clearer and more fluent. The human based evaluation involved ratings from 12 raters on a set of 30 input queries related to 9 knowledges bases. In comparison, the template based system generates one clause per relation and, on long queries, is judged unnatural (low fluency) by the raters. The symbolic generator often fails to adequately segment the input or to score the most fluent output highest. Figure 9 shows an example input and the corresponding output by each of the three systems being compared.

Input Query	Flight hasCurrentDepartureDate.[Date] hasCurrentArrivalDate.[Date] hasDestination.[Airport hasFlightTo.[Airport]] hasCarrier.[Airline] hasTicket.[AirTicket hasDateOfIssue.[Date]]
Temp	I am looking for a flight. Its current departure date should be a date. The current arrival date of the flight should be a date. The destination of the flight should be an airport. The airport should have flight to an airport. The carrier of the flight should be an airline. The ticket of the flight should be an air ticket. The air ticket should have date of a date.
Hyb	I am looking for a flight whose current departure date should be a date, whose current arrival date should be a date and whose destination should be an airport. The airport should have flight to an airport. The carrier of the flight should be an airline. The ticket of the flight should be an air ticket whose date of issue should be a date.
Symb	I am looking for a flight whose current departure date should be a date and whose current arrival date should be a date and whose destination should be an airport which should have flight to an airport. Its carrier should be an airline, the ticket of the flight should be an air ticket and its date of issue should be a date.

Fig. 9. Example input and outputs. Temp is a template based system, Symb the symbolic generator described in Section 2.3 and Hyb the same generator augmented with the Hypertagger

6 Conclusion

Syntax describes how words combine together to form complex NL expressions. Syntax is also often viewed as a scaffold for semantic construction. In other words, syntax provides both a means to abstract over lexical units and to mediate between form and meaning. While, when enough training data is available, statistical approaches can be developed which directly map meaning to form, there is, linguistically, no good reason to ignore the wealth of research that has gone into describing the syntax and the syntax/semantics interface of natural languages. In this paper, I have argues that syntax is in fact, a valuable compoment of natural language generation. In particular, I have shown that, by providing a

higher level of abstraction, syntax permits improving a hypertagger performance; facilitates the development of a generic, domain independent, query verbaliser; and supports the induction of compact, linguistically principled grammars which are well suited for data-to-text generation.

Acknowledgements

Thanks to Laura Perez-Beltrachini and Bikash Gyawali for working with me on grammar-based generation from knowledge bases; to Eva Banik, Eric Kow and Vinay Chaudry for organising the KBGen challenge; and to Enrico Franconi for his collaboration on generating queries from Knowledge Bases.

References

1. Angeli, G., Liang, P., Klein, D.: A simple domain-independent probabilistic approach to generation. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. pp. 502–512. Association for Computational Linguistics (2010)
2. Baldridge, J., Kruijff, G.J.M.: Multi-modal combinatory categorial grammar. In: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1. pp. 211–218. Association for Computational Linguistics (2003)
3. Bangalore, S., Rambow, O.: Using tag, a tree model, and a language model for generation. In: In Proceedings of the 1st International Natural Language Generation Conference. Citeseer (2000)
4. Banik, E., Gardent, C., Kow, E., et al.: The kbgen challenge. In: Proceedings of the 14th European Workshop on Natural Language Generation (ENLG). pp. 94–97 (2013)
5. Belz, A.: Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. Natural Language Engineering 14(4), 431–455 (2008)
6. Bos, J.: Predicate logic unplugged. In: Dekker, P., Stokhof, M. (eds.) Proceedings of the 10th Amsterdam Colloquium. pp. 133–142 (1995)
7. Brew, C.: Letting the cat out of the bag: Generation for shake-and-bake mt. In: Proceedings of the 14th conference on Computational linguistics-Volume 2. pp. 610–616. Association for Computational Linguistics (1992)
8. C. Gardent, B.G., Perez-Beltrachini, L.: Using regular tree grammar to enhance surface realisation. Natural Language Engineering 17, 185–201 (2011), special Issue on Finite State Methods and Models in Natural Language Processing
9. Cahill, A., Van Genabith, J.: Robust pcfg-based generation using automatically acquired lfg approximations. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics. pp. 1033–1040. Association for Computational Linguistics (2006)

10. Chen, D.L., Mooney, R.J.: Learning to sportscast: a test of grounded language acquisition. In: Proceedings of the 25th international conference on Machine learning. pp. 128–135. ACM (2008)
11. Coch, J.: Evaluating and comparing three text-production techniques. In: Proceedings of the 16th conference on Computational linguistics-Volume 1. pp. 249–254. Association for Computational Linguistics (1996)
12. Copestake, A., Lascarides, A., Flickinger, D.: An algebra for semantic construction in constraint-based grammars. In: Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics. Toulouse, France (2001)
13. Dahl, D.A., Bates, M., Brown, M., Fisher, W., Hunicke-Smith, K., Pallett, D., Pao, C., Rudnicky, A., Shriberg, E.: Expanding the scope of the atis task: The atis-3 corpus. In: Proceedings of the workshop on Human Language Technology. pp. 43–48. Association for Computational Linguistics (1994)
14. Dethlefs, N., Hastie, H., Cuayáhuitl, H., Lemon, O.: Conditional Random Fields for Responsive Surface Realisation using Global Features. Proceedings of ACL, Sofia, Bulgaria (2013)
15. Dongilli, P.: Natural language rendering of a conjunctive query. KRDB Research Centre Technical Report No. KRDB08-3). Bozen, IT: Free University of Bozen-Bolzano 2, 5 (2008)
16. Franconi, E., Guagliardo, P., Trevisan, M.: An intelligent query interface based on ontology navigation. In: Proceedings of the Workshop on Visual Interfaces to the Social and Semantic Web (VISSW 2010). vol. 565. Citeseer (2010)
17. Gardent, C., Kow, E.: A symbolic approach to near-deterministic surface realisation using tree adjoining grammar. In: ACL07 (2007)
18. Gardent, C., Kallmeyer, L.: Semantic construction in ftag. In: Proceedings of the 10th meeting of the European Chapter of the Association for Computational Linguistics. Budapest, Hungary (2003)
19. Gardent, C., Perez-Beltrachini, L.: Rtg based surface realisation for tag. In: COLING'10. Beijing, China (2010)
20. Gyawali, B., Gardent, C.: Surface realisation from knowledge-base. In: ACL. Baltimore, USA (June 2014)
21. Hockenmaier, J.: Data and models for statistical parsing with combinatory categorial grammar (2003)
22. Koller, A., Striegnitz, K.: Generation as dependency parsing. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. pp. 17–24. Association for Computational Linguistics (2002)
23. Konstas, I., Lapata, M.: Concept-to-text generation via discriminative reranking. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1. pp. 369–378. Association for Computational Linguistics (2012)
24. Konstas, I., Lapata, M.: Unsupervised concept-to-text generation with hypergraphs. In: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 752–761. Association for Computational Linguistics (2012)

25. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001)
26. Liang, P., Jordan, M.I., Klein, D.: Learning semantic correspondences with less supervision. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1. pp. 91–99. Association for Computational Linguistics (2009)
27. Lu, W., Ng, H.T., Lee, W.S.: Natural language generation with tree conditional random fields. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1. pp. 400–409. Association for Computational Linguistics (2009)
28. McCallum, A.K.: Mallet: A machine learning for language toolkit (2002)
29. Perez-Beltrachini, L., Gardent, C.: Hypertagging for query generation (May 2014), submitted
30. Perez-Beltrachini, L., Gardent, C., Franconi, E.: Incremental query generation. In: EACL. Gothenburg, Sweden (April 2014)
31. Portet, F., Reiter, E., Hunter, J., Sripada, S.: Automatic generation of textual summaries from neonatal intensive care data. In: Artificial Intelligence in Medicine, pp. 227–236. Springer (2007)
32. Reiter, E., Sripada, S., Hunter, J., Yu, J., Davy, I.: Choosing words in computer-generated weather forecasts. Artificial Intelligence 167(1), 137–169 (2005)
33. Steedman, M.: The syntactic process, vol. 35. MIT Press (2000)
34. The XTAG Research Group: A lexicalised tree adjoining grammar for english. Tech. rep., Institute for Research in Cognitive Science, University of Pennsylvannia (2001)
35. Trevisan, M.: A Portable Menuguided Natural Language Interface to Knowledge Bases for Querytool. Master's thesis, Free University of Bozen-Bolzano (Italy) and University of Groningen (Netherlands) (2010)
36. Vijay-Shanker, K., Joshi, A.K.: Feature structures based tree adjoining grammars. In: Proceedings of the 12th conference on Computational linguistics-Volume 2. pp. 714–719. Association for Computational Linguistics (1988)
37. Walter, S., Unger, C., Cimiano, P.: A corpus-based approach for the induction of ontology lexica. In: Natural Language Processing and Information Systems, pp. 102–113. Springer (2013)
38. White, M.: Efficient realization of coordinate structures in combinatory categorial grammar. Research on Language and Computation 4(1), 39–75 (2006)
39. Wong, Y.W., Mooney, R.J.: Generation by inverting a semantic parser that uses statistical machine translation. In: HLT-NAACL. pp. 172–179 (2007)