A Statistical, Grammar-Based Approach to Micro-Planning

Claire Gardent, Laura Perez-Beltrachini

A Statistical, Grammar-Based Approach to Micro-Planning

Claire Gardent
CNRS and Université de Lorraine
LORIA Campus Scientifique BP 239
F-54506 Vandoeuvre-lès-Nancy Cedex

Laura Perez-Beltrachini
CNRS and Université de Lorraine
LORIA Campus Scientifique BP 239
F-54506 Vandoeuvre-lès-Nancy Cedex

While there has been much work in recent years on data-driven natural language generation, little attention has been paid to the fine grained interactions that arise during micro-planning between aggregation, surface realization and sentence segmentation. In this paper, we propose a hybrid symbolic/statistical approach to jointly model these interactions. Our approach integrates a small hand-written grammar, a statistical hypertagger and a surface realization algorithm. It is applied to the verbalization of knowledge base queries and tested on 13 knowledge bases to demonstrate domain independence. We evaluate our approach in several ways. A quantitative analysis shows that the hybrid approach outperforms a purely symbolic approach in terms of both speed and coverage. Results from a human study indicate that users find the output of this hybrid statistic/symbolic system more fluent than both a template- and a purely symbolic grammar-based approach. Finally, we illustrate by means of examples that our approach can account for various factors impacting aggregation, sentence segmentation and surface realization.

1. Introduction

When generating a text, many choices must be made. The content to be expressed must be selected (*Content Selection*) and structured (*Document Planning*). Content must be distributed into sentences (*Sentence Segmentation*). Words (*Lexicalization*) and syntactic structures (*Surface Realization*) must be chosen. Appropriate referring expressions must be identified to describe entities (*Referring Expression Generation (REG)*). Coordinated and elliptical constructs may be exploited to omit repeated information (*Aggregation*).

These decisions interact and are subject to various constraints. Consider for instance the content sketched in Example (1). There are many ways of verbalising this content[1] but the appropriate verbalization depends on the context. For instance, the elision form (1l-m) is only appropriate in a context where another `sell` literal is present (e.g., `car(x) sell(x,y) sportsCar(y) sell(x,z) trucks(z)`). In this case, the repeated `sell` predicate can be elided (*A car dealer selling sports cars and trucks*).

(1) `CarDealer` $\sqcap \exists$`sell.(Truck)`
a. *The car dealer should sell trucks.* (Canonical Clause)
b. *It should sell trucks.* (Canonical Clause with Pronominal Subject)

1 As shall be discussed in Section 3, our approach was developed for generating user queries on knowledge bases. In this context, we choose to include the *should* modality in the verbalization of a binary relation thus capturing the intention of the user. Hence the unusual modal verbalizations. Nothing hinges on this though and, in our approach, generating the simpler non modal form (for example, *The car dealer sells trucks*) is a simple matter of modifying the grammar trees to remove the modal particle.

c. *and the car dealer should sell trucks* (And S-Coordination)
d. *and it$_0$ should sell trucks* (And S-Coordination with Pronominal Subject)
e. *The car dealer who should sell trucks* (Subject Relative)
f. *The car dealer (...) and who should sell trucks* (And Suject Relative)
g. *The car dealer (...), who should sell trucks* (Comma Subject Relative)
h. *The car dealer selling trucks* (Gerund)
i. *The car dealer (...) and selling trucks* (And Gerund)
j. *The car dealer (...), selling trucks* (Comma Gerund)
k. *trucks which the car dealer sells* (Object Relative Clause)
l. *The car dealer (selling ...) and trucks* (And NP)
m. *The car dealer (selling ...), trucks* (Comma NP)

The constraints regulating the choice of a given verbalization can be either soft or hard constraints. Thus a clause starting with a comma must be complemented by one starting with a coordination (Examples 2a-b) and an elided clause must follow its source clause (Examples 2c-d). These are hard, grammatical, constraints. Violating them yields sub-standard text.

(2) a. *The car dealer should sell trucks, provide sports cars and be located in France.*
b. ⋆ *The car dealer should sell trucks, provide sports cars, be located in France.*
c. *A car dealer selling trucks and sports cars*
d. ⋆ *A car dealer and sports cars selling trucks*

On the other hand many syntactic and linear ordering choices are regulated by soft constraints, that is, yield text of variable acceptability. Thus, although both sentences in (3) are grammatical, example (3a) is arguably better English than example (3b).

(3) a. *I am looking for a teaching assistant who is employed by the University, who teaches English and who has a PhD*
b. *? I am looking for a teaching assistant employed by the University, who teaches English and having a PhD*

In this article, we present a hybrid symbolic/statistical approach designed to handle the interactions between surface realization, sentence segmentation and aggregation. In this approach, hard constraints are encoded by the grammar (e.g., the constraints encoding the interactions beween comma and coordination conjunctions) while soft constraints (e.g., constraints on linear oder) are modeled statistically using a linear Conditional Random Field hypertagger.

To illustrate the workings of our approach, we consider an NLG setting where content selection and linearization are given namely, the verbalization of knowledge-base queries in the context of the Quelo Natural Language (NL) user interface to knowledge-bases (Franconi, Guagliardo, and Trevisan 2010b). For instance, given the query in (4a), we seek to generate a verbalization of this query such as (4b).

(4) a. `NewCar` $\sqcap \exists$`exteriorColor.Beige` $\sqcap \exists$`hasCarbody.UtilityVehicle` $\sqcap \exists$`runOn.NaturalGas` $\sqcap \exists$`locatedInCountry.Country`
b. *I am looking for a new car whose exterior color should be beige and whose body style should be a utility vehicle. The new car should run on natural gas and should be located in a country.*

We compare our approach both with a template- and with a symbolic, grammar-based approach and show that it improves performance in terms of both speed and output quality.

One distinctive feature of our method is that it is grammar based. As expected, this allows for a detailed handling of syntactic and morpho-syntactic constraints (e.g, subject/verb agreement, verb tense, relative pronoun case). More interestingly, this also allows for the training of a "high level hypertagger" whose categories are not lexical or syntactic categories but general, more abstract, syntactic classes describing the surface realization of for instance, a verb argument. This contrasts both with approaches to data-to-text generation which map meaning representations to sentences without assuming an intervening syntax (Konstas and Lapata 2012b, 2012a; Lu, Ng, and Lee 2009; Dethlefs et al. 2013); and with traditional supertagging approaches which operate on lexical categories thereby requiring a large training corpus (Bangalore and Joshi 1999; Espinosa, White, and Mehay 2008).

Another interesting feature of our approach is that it is domain independent and can be applied to any Knowledge Base independent of its domain. As we shall show in Section 4.3, because it relies on a generic grammar, an automatically induced lexicon and a hypertagger trained on a small data-to-text corpus, our approach can be applied to any knowledge-base independent of the domain it covers.

In sum, the main features of our approach to query generation are that:

- it jointly models sentence segmentation, aggregation and surface realization (cf. Section 5 and 6)
- the grammar based approach provides abstract syntactic classes which capture linguistic generalizations (e.g., subject relative clause) thereby allowing for learning with little training data (cf. Section 3 and 4)
- it is domain independent and does not require additional parallel data/text training corpus for porting to a new knowledge base (cf. Section 5)

The article is structured as follows. Section 2 summarizes related work on joint models for micro-planning. Section 3 describes the query generation task and the NLG architecture we developed. Sections 4 and 5 present the experimental setup used for the evaluation and the results obtained. Section 6 provides a qualitative analysis of the generated output by showing examples of interactions between aggregation, sentence segmentation and surface realization that were correctly accounted for by our approach. Section 7 concludes with pointers for further research.

2. Related work

Earlier rule-based work on micro-planning NLG has explored various ways of combining lexicalization, surface realization and aggregation in architectures ranging from integrated systems where all decisions are made simultaneously (Appelt 1982) to strictly sequential pipelines (Reiter, Dale, and Feng 2000). While the sequential approach is easier to develop and to maintain, it cannot easily account for the interactions that are known to exist between the various modules (Danlos 1987). A sequential approach can in fact induce a "generation gap" (Meteer 1990) whereby generation fails because a choice made earlier in the pipeline conflicts with the constraints of a module occurring further down the pipeline. Moreover, taking individual decisions at different sub-tasks in a sequential manner might lead to suboptimal solutions (Marciniak and Strube 2005). On the other hand, symbolic joint approaches to microplanning lack in robustness and efficiency and require much time and expertise to develop the various linguistic resources (grammar, lexicon, text plans, etc.) they are based upon.

In previous work on sentence planning, Walker, Rambow, and Rogati (2001) proposed a trainable sentence planner, namely SPoT, for a dialog system in the travel domain which was later on extended to provide restaurant information (SPaRKy, (Walker et al. 2007)). Their approach addresses the interactions occurring between content ordering, lexicalization and aggregation. Each input dialog act is assigned a syntactic structure (DSyntS, Deep Syntactic Structure (Mel'čuk 1988)) and then alternative ways of combining them into one or several sentences are explored. To this end SPoT proceeds in two steps. First, a number of random alternative sentence plans is generated using a set of clause combining operations and hand-crafted heuristics. Second, a ranking function learnt from a corpus of sentence plans annotated with human ratings is applied to score the sentence plans generated in the first step. SPaRKy is based on SPoT's two steps sentence plan generation and ranking approach, but it incorporates rhetorical structure in the generated sentence plans. The major differences with our approach are the following. First, their sentence planner generates different orderings of dialog acts while in our case the order is enforced by the query linearization. Second, while they use a set of aggregation operations to specify clause-combinations, in our approach aggregation is modeled by our grammar. Finally, the ranking-based approach is able to take into account global view on the sentence plans looking to wider evidence; however, the possibilities to find an optimal solution rely on the capabilities of the first sentence plan generation step.

Recently, several joint data-driven approaches have been proposed which aim to account for the multi-way interactions between various NLG modules while minimizing the amount of expertise and manual work required.

(Konstas and Lapata 2012b, 2012a) departs from the sequential, statistical approach proposed by Angeli, Liang, and Klein (2010) for generation from databases and describes a generation model which jointly performs content selection, sentence planning and surface realization. Given a corpus of database records and their textual descriptions, they induce a probabilistic context free grammar which captures the structure of the database and how it can be rendered into natural language. Generation then boils down to finding the best parse tree using the Viterbi algorithm. They evaluate their approach on three domains and obtain results competitive with the state of the art. (Konstas and Lapata 2012b, 2012a; Lu, Ng, and Lee 2009; Dethlefs et al. 2013) developed NLG systems which are trained on parallel corpora of text and databases such as Geoquery (880 training instances, queries of a geographic database), Robocup (1539 instances, coaching advice to robots), WeatherGov (29 528 instances, weather forecasts) and ATIS (5426 instances, air travel). Dethlefs et al. (2013) train their model on a corpus of restaurant recommendations.

While these approaches often handle formal languages (e.g., sets of database records) and applications (e.g., coaching a robot or querying a geography database) that require handling a larger fragment of natural language than the simple language of entity description we focus on in this paper, the proposed approaches differ in two main ways from our approach. First, there is no systematic exploration of how aggregation and syntactic choices impact readability. While our hand-written grammar systematically captures the possible syntactic realizations of a given predicate, the probabilistic grammar acquired by Konstas and Lapata (2012b, 2012a) for instance will only encode the possible syntactic realizations of an input that can be learned from the training corpus. Second and more importantly, in all these approaches, the learned models are corpus specific and adaptation to a new domain requires the construction of a new parallel corpus of meaning representations and natural language sentences. Konstas and Lapata's approach (2012b, 2012a) makes use of relatively large training corpora with respectively 1539, 29528 and 5426 input/output pairs for each of the three domains considered. In contrast, we use 206 input/output pairs to train a hypertagging module which, together with a small hand-written grammar and an automatically induced lexicon, permits generating from arbitrary knowledge bases.

Zarrieß and Kuhn (2013) consider referring expressions, syntax and word order and explore how different architectural setups account for their interactions. Using a corpus annotated with deep syntax and discourse referents, they develop a statistical approach which can map a deep syntax tree and a set of referents to a sentence. The approach combines a syntax generator mapping a deep to a shallow dependency tree, a referring expression generator and a linearizer. They combine these three modules in different ways and examine how these different combination modes impact the generated text.

As in (Konstas and Lapata 2012b, 2012a), in Zarrieß and Kuhn's approach (2013), the syntactic variations allowed for a given input are restricted to those learned from the parallel corpus of deep and shallow syntax. There is, for instance, no mapping from repeated or shared content to elided constructions; or to relative clauses. More generally, while our grammar systematically encodes the various ways in which a proposition can be verbalized (e.g., using a relative clause, an elided clause etc.) and uses these to support aggregation, Zarrieß and Kuhn (2013) use a limited set of learned transformations to map deep to shallow syntax. Empirically, another difference with our work is that while Zarrieß and Kuhn (2013) focus on the interactions between referring expressions, syntax and word order, we work on the interactions between surface realization, aggregation and sentence segmentation.

Lampouras and Androutsopoulos (2013) present a joint model for content selection, surface realization and aggregation. Using Integer Linear Programming, they specify constraints designed to maximize the importance and the number of the selected facts so as to enhance informativeness while minimising the number of selected entities to favor aggregation. They apply their approach to the task of verbalising sets of OWL axioms and show that in comparison to a handcrafted NLG system, their approach provides more compact text with no deterioration in text quality.

This approach is similar to ours in that it focuses on modeling the interactions between aggregation and surface realization. There are two main differences however. A first main difference is that we model surface realization and aggregation using a grammar. Language naturally allows for aggregation. Relative clauses, shared subject construction, ellipsis, coordination are all means of factoring out common content. By using a grammar which describes these phenomena, we directly account for the interaction between surface realization and aggregation. In contrast, Lampouras and Androutsopoulos (2013) make use of word specific sentence plans for surface realization and of ad hoc sentence plan combining rules for aggregation. A second difference is that while, in our approach, syntax and aggregation choices are guided by a hypertagger trained to predict the best sequence of syntactic constructs for a given input, in (Lampouras and Androutsopoulos 2013), the aim is to systematically minimize the length of the output, that is, to maximize aggregation. That is, while we allow for various ways of aggregating a given content into different sentences and select one based on linguistic and semantic criteria, Lampouras and Androutsopoulos (2013) select the aggregated sentences based solely on sentence length.

In sum, our approach differs from previous work in two main ways. First, it focuses on providing a joint model for the interactions between surface realization, aggregation and sentence segmentation. In contrast, previous joint approaches have focused on the interactions between: Content selection, sentence planning and surface realization (Konstas and Lapata 2012b, 2012a); referring expressions, syntax and word order (Zarrieß and Kuhn 2013); or content selection, lexicalization and aggregation(Zarrieß and Kuhn 2013). Second, this joint model is based on a generic grammar which systematically captures the possible syntactic realizations of a proposition. In contrast, previous approaches only account for some of the possible syntactic variations using ad hoc templates (Lampouras and Androutsopoulos 2013) or transformation rules learned from annotated corpora (Zarrieß and Kuhn 2013).

3. Grammar based query generation

We start by defining the generation task (Section 3.1) and the semantic input it starts from (Section 3.2). We then describe the architecture of our generator (Section 3.3).

3.1 The Generation Task

In natural language (NL) interfaces to Knowledge Bases (KB), Natural Language Generation has been shown to successfully assist the user by allowing her to formulate a KB query while knowing neither the formal query language nor the content of the KB being queried (Franconi, Guagliardo, and Trevisan 2010a; Franconi et al. 2011; Franconi, Guagliardo, and Trevisan 2010b; Franconi et al. 2011). This is because, when using a natural language interface to Knowledge Bases, the user never sees the formal query. Instead, at each step in the query process, the generator verbalizes all extensions of the current query which are computed by the reasoning system to be plausible extensions of this query given the knowledge base under consideration[2]. The user then chooses from among the set of generated NL queries the query she intends. She can also modify the current query by adding, deleting or substituting content.

In practice, the user query is specified in an interactive process as follows. The system starts by proposing an empty query q_0 and a set of possible query extensions $q_1^1 \dots q_1^n$. The user then chooses one of the proposed extensions (q_1^i with $1 \leq i \leq n$) which triggers another proposition by the system of a set of possible extensions $q_2^1 \dots q_2^n$ given q_1^i. At each step in the query specification process, the system displays, not the formal query, but its natural language verbalization as produced from the formal query by the NLG engine. The following shows an example sequence of interactions which leads to the specification of the query $MarriedMan$. The formal language used to represent queries is that of conjunctive tree shaped queries and is defined in the following section.

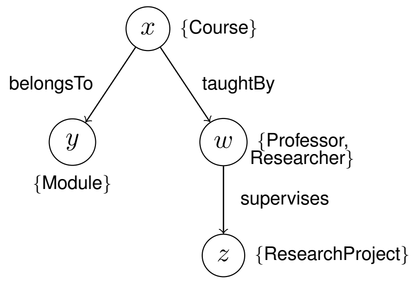
(5) a. *I am looking for **something*** (initial query)
`⊤`
b. *I am looking for **a man*** (substitute concept)
`Man`
c. *I am looking for a **young** man* (add compatible concept)
`Man ⊓ Young`
d. *I am looking for a young man **who is married to a person*** (add relation)
`Man ⊓ Young ⊓∃isMarried.(Person)`
e. *I am looking for a young **married man*** (substitute selection)
`MarriedMan ⊓ Young`
f. *I am looking for a married man* (delete concept)
`MarriedMan`

2 See (Franconi et al. 2011; Perez-Beltrachini, Gardent, and Franconi 2014) for a more detailed description of how these extensions are computed.

3.2 The Generation Input

Following (Franconi, Guagliardo, and Trevisan 2010a; Franconi et al. 2011; Franconi, Guagliardo, and Trevisan 2010b; Franconi et al. 2011), we assume a formal language for queries that supports the querying of various knowledge and data bases independently of their specification language. This language, called the language of tree shaped conjunctive queries, is a minimal query language that is shared by most knowledge representation languages and is supported by Description Logic (DL) reasoners. Specifically, the Query Tool formal framework (Guagliardo 2009) defines a *tree-shaped conjunctive query* as a labeled tree whose edges are labeled with relations and whose nodes are labeled with a variable and a non-empty set of concept names. Each node of the query tree can be expressed as a concept of a Description Logic (DL) $\mathcal{L}$ using atomic concept instantiation, existential restriction and conjunction. Given a knowledge base $\mathcal{K}$ over a set of relations **R** and a set of concepts **C**, a concept in $\mathcal{L}$ is defined as $S \ ::= C \ \mid \exists R.(S) \mid S \sqcap S$ where $R \in$ **R**, $C \in$ **C**, $\sqcap$ denotes conjunction and $\exists$ is used for limited existential restrictions.

Figure 1 shows an example query tree together with the concept associated with its root node.



Course $\sqcap$ $\exists$belongsTo.Module $\sqcap$ $\exists$taughtBy. (Professor $\sqcap$ Researcher $\sqcap$ $\exists$supervises.ResearchProject)

Figure 1
Example of query tree.

Informally, the input query is a directed tree, where each edge is labeled with exactly one binary predicate and each node is labeled with one or more unary predicates. Such a tree encodes a first-order logic query in which the root node represents a free variable; each other node represents a distinct, existentially quantified variable; the label R of an edge directed from node x to node y represents a formula R(x, y); each label C of a node x represents a formula C(x).

While tree shaped conjunctive queries allow for efficient reasoning, their limited expressivity restricts the range of semantic and linguistic phenomena which can be covered. No negations, no disjunctions, no universal quantifications may appear. Moreover, it is tree-shaped hence, no variables may appear twice as the second argument of a binary predicate. In practice, the natural language fragment that can be generated from such input is restricted to those cases where there is no coreference between the second argument of two binary relations (e.g., *John hates and Peter likes the new car*), no universal quantification (e.g., *All yogi are vegetarian*) and no negation or modality (e.g., *Not all yogi are vegetarian, Most yogi are vegetarian, Yogi might be vegetarian*).

As mentioned in the introduction, during natural language generation, document planning structures and orders the input that will be passed on to the micro-planning stage. In the context of the Quelo natural language interface to knowledge bases, document planning consists in linearizing the tree shaped conjunctive query which forms the input to surface realization in such a way that this linearization matches the order in which the user specified her query. This is enforced by first, using the order in which the user applies the query update operations (add, substitute, delete) to induce an order on the tree shaped query (e.g., if a relation r_1 is added by the user after a relation r_2, the edge labeled with r_1 will appear to the left of the edge labeled with r_2 in the query tree) and second, traversing the resulting tree in a depth-first, left-to-right fashion[3] .

The motivation for this particular choice of linearization is that, for cognitive reasons, the Natural Language query generated by the system should deviate as little as possible from the order in which the query is being built by the user. By constraining the linearization of this formal query to match the order in which the user formulates her query, the system provides a linearization information that can then be used by the surface realizer to adequately constrain the word order of the generated natural language query. Note that these two steps (linearization of the input and surface realization) are independent of each other. While the input to surface realization is ordered, it is still possible to generate a sentence whose word order does not match the order of the input. For instance, given the input `Course ⊓ ∃taughtBy.(Professor)`, our surface realizer can generate both the active (*The course should be taught by a professor*) and the passive (*A professor should teach the course*). To favor generated sentences whose surface order matches the order of the linearized input, we use a customized scoring function which computes a word order cost capturing the deviation between the input and the generated sentence order[4].

To generate from a tree shaped conjunctive query, we first linearize the query as described above (i.e., using a depth-first, left-to-right traversal of the tree and the precedence order defined on tree nodes and concept labels by (Dongilli 2008; Guagliardo 2009)). For instance, the tree shaped query shown in Figure 1 is linearized as shown in (6a).

We furthermore map this linearized formula to the format expected by our surface realizer by making explicit the arguments of concepts and relations using variables. For instance, (6a) is mapped to (6b).

(6) a. `Course ⊓ ∃belongsTo.Module ⊓ ∃taughtBy.(Professor ⊓ Researcher ⊓ ∃supervise.ResearchProject)`

b. *{Course(x), belongsTo(e1, x, y), Module(y), taughtBy(e2,x,w), Professor(w), Researcher(w), supervise(e3, w, z), ResearchProject(z)}*

3.3 The Generation Architecture

Our generation system consists of four modules:

- An automatically derived lexicon which associates relations and concepts with lexicalized grammatical structures;

3 See (Franconi, Guagliardo, and Trevisan 2010a) for a formal definition of the strict total order jointly imposed on the input query by the user operations and by the tree traversal.

4 See (Perez-Beltrachini, Gardent, and Franconi 2014) for more details on this scoring function.

- A symbolic, hand-written grammar which specifies these grammatical structures and encodes hard grammaticality constraints;
- A statistically trained hypertagger which filters the initial search space of the generator by applying soft statistical constraints learned from a small parallel data-to-text corpus;
- A surface realization algorithm which generates the space of possible outcomes licenced by the lexicon, the hypertagger, the grammar and a given input.

Semantics: `equippedWith`
Tree: nx0VVVpnx1
Syntax: Canonical
Anchor: *equipped*
Coanchor: V1 → *should/V*
Coanchor: V2 → *be/V*
Coanchor: P → *with/P*

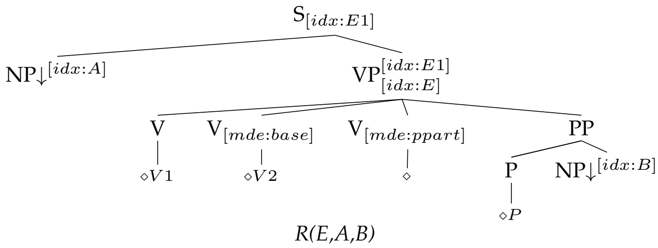


Figure 2
FB-LTAG tree and Lexical Entry for the relation `equippedWith`. Lexical selection "anchors" the anchor node (marked with ⋄) of the TAG tree with the anchor specified by the lexical entry (here, *equipped*) and instantiates its predicate variable R to its semantics (here `equippedWith`). Coanchors label the terminals named with the corresponding names (e.g., *should* labels the V node called $V1$).

3.3.1 Lexicon. The lexicon and the grammar describe the possible lexicalizations and surface realizations of KB concepts and relations. Figure 2 shows an example lexical entry and the corresponding grammar unit.

Lexical entries relate KB relations (here the `equippedWith` relation) and words (here the (co)-anchors, *should, be* and *equipped with*) to grammar units, that is, to trees and semantic schemas (here the right hand side of Figure 2). During generation, the relation is used to instantiate the predicate variable R in the semantic schema $R(E, A, B)$ and the Anchor value (*equipped*) to anchor the tree, that is, to label the terminal node marked with the anchor sign (⋄). Similarly, each Coanchor equation will be used to label the terminal node with corresponding name. For example, the strings *should, be* and *with* will be used to label the terminal nodes names $V1, V2$ and P respectively.

As mentioned in the introduction, we automatically derive lexicons from knowledge bases using the approach described in (Trevisan 2010). In brief, this approach consists in tokenizing and part-of-speech tagging relation and concept names with a customized tokenizer and part-of-speech (PoS) tagger. A set of hand-defined mappings is then used to map PoS sequences to TAG trees. The resulting lexicon maps the concepts and relations of each input KB to one or more grammar units (pair of semantic and tree schema), each unit capturing a possible lexical and/or syntactic verbalization of the corresponding concept/relation. For instance, for the relation `equippedWith`, the lexicon extraction procedure will create 16 lexical entries each corresponding to a mapping of the `equippedWith` relation to a different syntactic verbalization. Examples of these verbalizations and the corresponding tree names are shown in Table 1 (first two columns).

When tested on a corpus of 200 ontologies, this approach was shown by Trevisan (2010) to provide appropriate verbalization 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 verbalize these relation identifiers (see (Trevisan 2010) for more details on this evaluation).

Thus in general, the lexicon extraction method proposed by Trevisan (2010) provides a generic procedure for automatically lexicalising ontological data. While more sophisticated methods could be used to improve both coverage and output quality, we focus here on the interactions between surface realization, sentence segmentation and aggregation (rather than lexicalization) and leave the question of a better and more complete lexicalization method for further research.

3.3.2 Grammar. Following (Gardent and Kow 2007) we use a Feature-Based Lexicalized 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 (Vijay-Shanker and Joshi 1988). In essence, an FB-LTAG is a set of elementary trees whose nodes are decorated with feature structures and which can be combined using either substitution or adjunction to produce phrase structure trees (also called derived trees). 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 TAG, each derived tree is described by a unique derivation tree which records the elementary trees involved in the constrution of this tree together with the combining operations applied.

As illustrated in Figure 2, in an FB-LTAG with unification semantics, each tree is associated with a semantics, while shared variables between syntax and semantics ensure the correct mapping between syntactic and semantic arguments. When trees are combined, the semantics of the resulting derived tree is the union of their semantics modulo unification.

Figure 3 shows an example toy FB-LTAG with unification semantics. The dotted arrows indicate possible tree combinations (substitution for *car*, adjunction for *coupé*). As the trees are combined, the semantics is the union of their semantics modulo unification. Thus, given the grammar and the derivation shown, the semantics of *It sells a car, a coupé.* is as shown, namely `sell(a,d,c),car(c),coupe(c)` or equivalently $\exists$sell.(Car $\sqcap$ Coupe).

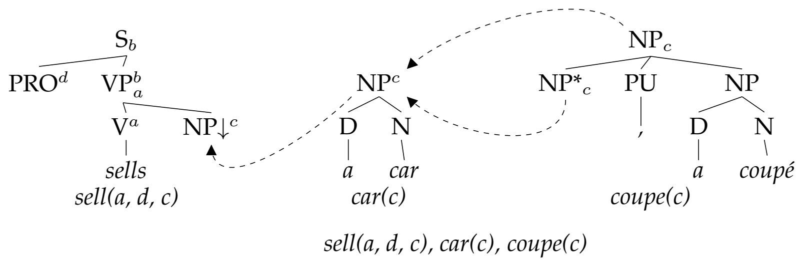


Figure 3
Derivation and semantics for "It sells a car, a coupé."

3.3.3 Chart-Based Surface Realization. For surface realization, we combine the chart-based algorithm described in (Gardent and Perez-Beltrachini 2010; Perez-Beltrachini, Gardent, and Franconi 2014) with a hypertagger filtering the initial search space. This algorithm proceeds in five main steps as follows.

- Given the input linearized query, hypertagging predicts n best sequences of grammar units. These grammar units are either FB-LTAG trees from the grammar or more abstract syntactic classes such as *subject relative* (SubjRel) or any of the syntactic classes listed in the rightmost column of Table 1.
- Lexical Selection retrieves from the grammar all lexical entries whose semantics subsumes the input semantics and which are consistent with the hypertagger filter. The grammar trees selected by these lexical entries are grounded with both the lexical and the semantic information contained in these entries.
- 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.
- Sentence Extraction: All syntactically complete trees which are rooted in S and are associated with exactly the input semantics are retrieved. Their yields provide the set of generated (lemmatized) sentences.
- Morphological Realization: Lexical lookup and unification of the features associated with lemmas in the generated lemmatized sentences yields the final set of output sentences.

For instance, given the linearized query in (7a), hypertagging might yield the two best sequences of hypertags (TAG tree names) shown in (7b). Given this, lexical selection will select the three trees shown in Figure 3 plus the relative clause tree betanxBEnx for the relation symbol `coupe`. Tree combination will then produce two complete phrase structure trees whose yield will be the lemmatized sentences *PRO sell a car, a coupé* and *PRO sell a car which be a coupé*. Morphological realization will transform these lemmatized sentences into *It sells a car, a coupé* and *It sells a car which is a coupé*.

(7) a. `∃sell.(Car ⊓ Coupe)`
b. PRO0VVnx1 nx betanxPUnx
PRO0VVnx1 nx betanxBEnx
c. *sell(a, d, c), car(c), coupe(c)*

3.3.4 Hypertagging. Super- and Hyper-tagging (Espinosa, White, and Mehay 2008) are preprocessing steps to parsing and surface realization which assign likely categories to the input based on contextual information. Supertagging was first introduced by Bangalore and Rambow (2000) to assign likely categories to words before parsing starts thereby reducing the initial search space. They showed that supertagging speeds up parsing times considerably. Likewise (Curran, Clark, and Vadas 2006; Clark and Curran 2004) showed that supertagging lead to extremely efficient CCG (Combinatory Categorial Grammar) parsing and Espinosa, White, and Mehay (2008) that hypertagging can achieve substantial improvements in realization speed with superior realization quality.

Similarly, we use hypertagging to improve efficiency. Importantly however, we also use hypertagging to monitor several of the choices that need to be made during the microplanning stage of generation.

Contrary to parsing where supertagging aims to identify a single correct sequence of parts-of-speech tags for the input string, in surface realization, there may be several sequences of grammar units which all lead to correct output sentences. However, these sentences may be more or less fluent. In our approach, hypertagging helps predict the

Table 1
Verbalizations of the `equippedWith` relation captured by the lexicon and the grammar. The second column lists the corresponding tree names while the third shows the corresponding syntactic class.

Verbalization pattern	Tree	Synt.Cl
NP_0 should be equipped with NP_1	sDOTnx0VVVpnx1	Canonical
It_0 should be equipped with NP_1	sDOTPRO0VVVpnx1	Canonical
and NP_0 should be equipped with NP_1	sCONJnx0VVVpnx1	S-Coordination
and it_0 should be equipped with NP_1	sCONJPRO0VVVpnx1	S-Coordination
NP_0 which should be equipped with NP_1	W0nx0VVVpnx1	SubjRel
NP_0 (...) and which should be equipped with NP_1	ANDWHnx0VVVpnx1	SubjRelPU
NP_0 (...), which should be equipped with NP_1	COMMAWHnx0VVVpnx1	SubjRelPU
NP_0 equipped with NP_1	betanx0VPpnx1	PpartOrGerundOrPrerp
NP_0 (...) and equipped with NP_1	betanx0ANDVPpnx1	SharedSubj
NP_0 (...), equipped with NP_1	betanx0COMMAVPpnx1	SharedSubj
NP_1 with which NP_0 should be equipped	W1pnx1nx0VV	PObjRel
NP_0 (equipped with X) and with NP_1	betavx0ANDVVVpnx1	Ellipsis
NP_0 (equipped with X), with NP_1	betavx0COMMAVVVpnx1	Ellipsis

sequences of grammar trees which yield the most fluent sentences. It helps decide when to use an ellipsis or a coordination (aggregation); how to distribute the input data into clauses and sentences (sentence segmentation) and which syntactic form to use for a given relation in a given context (surface realization).

How does this work? As illustrated in Table 1, the trees of a Tree Adjoining Grammar provide a detailed specification of both the lexicalization and the syntactic constructions licensed by a given semantic literal. For instance, the nx0VVnx1 tree describes the syntactic structure of a transitive verb occurring in a canonical clause (e.g., *The car dealer should sell trucks*), the nx0VVpnx1 tree specifies a canonical clause containing a verb taking a prepositional complement (e.g., *The car should run on fuel*) and the W0nx0VVnx1 tree captures a transitive verb occurring in a subject relative clause (e.g., *The car dealer which should sell trucks*). In effect, each TAG tree embodies one or more microplanning decisions. For instance, selecting an sDOTnx0VVVpnx1 or sDOTPRO0VVVpnx1 tree licences the beginning of a new sentence while selecting an SCONJnx0VVVpnx1 tree induces a sentence coordination. Ellipses result from using, for example, the betavx0ANDVVVpnx1 or betavx0ANDVVVpnx1 tree while selecting an W0nx0VVVpnx1, COMMAW0nx0VVVpnx1 or ANDW0nx0VVVpnx1 tree yields a relative clause.

To favor sequences of TAG trees which result in fluent, natural sounding verbalizations of KB queries, we train a CRF model on a small corpus of aligned formal KB queries and sequences of TAG trees or of syntactic classes. Indeed, we experiment with two models: One that predicts TAG trees (e.g., W0nx0VVVpnx1) and another which predicts more abstract grammatical classes (e.g., Relative Clause). For example, given the query shown in (8a), the first model will be trained on the tree annotations shown in (8b) while the second will be trained on the syntactic classes annotations shown in (8c).

(8) a. `CarDealer ⊓ ∃locatedIn.(City ⊓ ∃sell.(Car ⊓ ∃runOn.Diesel))`
b. (Trees) Tnx Tbetanx0VPpnx1 Tnx TANDWHnx0VVnx1 Tnx Tnx0VVpnx1 Tnx
c. (Synt.Classes) NP ParticipialOrGerund NP SubjRelPU 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.*

The tags learned by the hypertagger are therefore either tree names or more general *syntactic classes* which capture the syntactic realization of a semantic token independent of its lexical class. We use a set of 10 syntactic classes. Most of them are illustrated in Table 1, namely, new clause, conjoined sentential clause, subject relative clause with and without coordination, participial, gerund and prepositional phrase construction, shared subject construction and ellipsis. Three additional syntactic classes not illustrated in Table 1 are adjective modifiers, noun or adjective arguments and apposition. These syntactic classes are automatically associated by the grammar compiler used to compile the FB-LTAG described in Section 4.2 with each of its trees.

During the lexical selection step, only those TAG trees which are compatible with the hypertagger predictions will be retrieved and added to the chart. For instance, given the KB symbol `equippedWith`, while lexical selection will return the set of trees shown in Table 1, if the hypertagger predicts the SubjRelPU class for this literal, then the tree combination step of the generation algorithm will only consider the trees labeled with that syntactic class. In this way, the hypertagger makes high-level microplanning decisions while the grammar and the lexicon further refine those decisions by enforcing hard constraints such as the possible subcategorization pattern of a given literal (encoded in the lexicon), for example sDOTnx0VVVpnx1, or the choice (encoded in the grammar trees) between a comma-conjoined relative clause (COMMAW0nx0VVVpnx1) and a relative clause introduced by *and* (ANDW0nx0VVVpnx1). In other words, we use the hypertagger to rank the syntactic construct sequences in terms of naturalness while the grammar and the lexicon are used to enforce hard lexical and grammatical constraints such as the fact that a comma-separated clause must be followed by a clause introduced by a coordination word (e.g., *and* or *or*).

Table 2
Example Canonical Sentences and Associated Subcategorization Classes all mapping to the "Canonical" syntactic class .

Verbalization pattern	Tree	Synt.Cl
NP_0 should generate NP_1	sDOTnx0VVnx1	Canonical
NP_0 should run on NP_1	sDOTnx0VVpnx1	Canonical
NP_0 should be equipped with NP_1	sDOTnx0VVVpnx1	Canonical
NP_0 should be the equipment of NP_1	sDOTnx0VVDNpnx1	Canonical
NP_0 should have access to NP_1	sDOTnx0VVNpnx1	Canonical
NP_0 should be relevant to NP_1	sDOTnx0VVApnx1	Canonical
NP_0 should be an N_1 product	sDOTnx0VVDNnx1	Canonical
NP_0 with NP_1	betanx0Pnx1	Canonical

4. Experimental Setup

We developed and tested the generation approach described in the preceding section on 13 knowledge bases namely, two ontologies on Cars and on Master courses developed by the Quelo consortium and 11 ontologies available on the web including the

Aquatic Resource Observation ontology, the GoodRelations ontology, Wines, QALL-ME (Ferrandez et al. 2011), Adolena Ontology (Keet et al. 2008), Movies, The Air System Ontology (TONES repository), Camera OWL Ontology and Travel (Protégé repository), The Photography Ontology, and The Bibliographic Ontology.

This involved automatically acquiring lexicons from these knowledge bases; manually specifying a Feature-Based Lexicalized Tree Adjoining Grammar describing the morpho-syntax, the syntax and the semantics of KB queries; developing a parallel corpus of formal and natural language KB queries to train the hypertagger model; training the hypertagger model on that corpus and integrating this hypertagger with the surface realization algorithm described in (Perez-Beltrachini, Gardent, and Franconi 2014).

4.1 Automatic Induction of Lexicons

Our lexicon is automatically derived from 13 knowledge bases. It includes 10020 lexical entries for 1296 concepts and relations and has an average lexical ambiguity rate (number of lexical entries per KB symbol) of 7.73.

4.2 Hand-Written Grammar

We manually developed a Feature-Based Lexicalized Tree Adjoining Grammar using the XMG grammar writing formalism (Crabbé et al. 2013). The grammar consists of 135 trees describing canonical and non canonical surface forms for relations and concepts. Canonical surface realizations are illustrated in Table 2. Non canonical variants include finite clauses with pronominal subject, coordinated sentences and coordinated VPs, subject, object and pied piping relative clauses, participials and gerund, verbal ellipsis and prepositional phrases (cf. Table 1).

In essence, because it captures the syntax of KB queries, the grammar describes the language of entity descriptions. A KB query identifies a set of objects by specifying properties (concepts) of these objects and of other objects these objects are related to. Thus verbalizations of KB queries are in effect descriptions of objects or sets of objects which involves chaining unary and binary relations to describe the set of objects the user wants to identify. Examples of the NL queries our system generates are shown in Section 6.

4.3 Hypertagger

We view hypertagging as a sequence labeling task in which a sequence of KB symbols needs to be labeled with appropriate syntactic labels[5]. In practice, we learn a linear-chain Conditional Random Field (CRF, (Lafferty, McCallum, and Pereira 2001)) model to predict the mapping between observed input features and hidden syntactic labels. This probabilistic model defines the posterior probability of syntactic labels y=$\{y_1, \ldots, y_L\}$

5 Recall that the linear order of the semantic input is deterministically given by the linearization process of the tree based conjunctive input (cf. Section 3.2).

given the sequence of input literals x=$\{x_1, \ldots, x_n\}$:

$$P(y \mid x) = \frac{1}{Z(x)} \prod_{l=1}^{L} exp \sum_{k=1}^{K} \theta_k \Phi_k(y_l, y_{l-1}, x)$$

Z(x) is a normalization factor and the parameters θ_k are weights for the feature functions Φ_k. Feature functions are defined over the entire input semantics x, the previous label (y_{l-1}) and the current syntactic label (y_l).

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

$$y^* = argmax_y P(y \mid x)$$

The most likely hypertag sequence is computed using the Viterbi algorithm. We used the Mallet toolkit (McCallum 2002) for parameter learning and inference.

4.3.1 Training Corpus. To train the CRF, we constructed a corpus aligning formal queries with sequences of syntactic labels, either TAG trees or syntactic classes, as shown in Example (8). The list of TAG trees and syntactic classes used for annotation is shown in Appendix A.

We created a data set of 206 training instances semi-automatically as follows.

First, we manually created input semantics (i.e., tree shaped conjunctive queries) for 11 ontologies for different domains[6] taking care to include query patterns illustrating different lexicalization, segmentation, aggregation and surface realization possibilities. These patterns vary in terms of length[7] (min: 2, max: 19, avg: 7.44) and of query tree shape (maximum depth: 4 and maximum fanout: 6). To capture the impact of lexicalization on micro-planning, we additionally make sure to include various types of KB relation symbols using the classification of relations mentioned in section 3.3.1. This L(exicalization)-Classification is defined in (Trevisan 2010) and consists of 13 classes (henceforth, L-Classes). In essence, it provides an abstract characterization of the lexicalization pattern of a KB relation[8]. For instance, the relation `equippedWith` is associated with the class *VBN-Be* because it can be verbalized as *NP_0 should **be equipped with** NP_1* whereas the relation `scientificName` will be associated with the class *Simple-NP* because it can be verbalized as *The **scientific name of** NP_0 should be NP_1*. More generally, relation symbols belonging to different classes will induce different lexicalizations and thereby have a different impact on surface realization. By including KB symbols from different classes, we therefore create a training corpus that integrates variation not only in terms of the length and the shape of the input but also in terms of the lexicalizations that are possible for the KB symbols.

Using the set of input semantics specified as described above, we then generated query verbalizations from these queries using semi-automatically defined microplans and the symbolic surface realizer described in (Perez-Beltrachini, Gardent, and Franconi

6 The domains covered by the 11 ontologies are all those enumerated at the beginning of Section 4 except for The Air System and The Bibliographic Ontologies.
7 Length is defined as the number of KB concepts and relations.
8 The training set covers 12 of these 13 classes as the L-Class *VBG* is not present in the corpus.

2014). The microplans indicate the segmentation of the query. In some cases they also include lexicalization choices for some elements of the query. The surface realizer uses the same grammar, lexicon and surface realizer as the approach described here but does not integrate the hypertagger. This symbolic approach to microplanning yielded a total of 6841 outputs which we disambiguated manually, choosing for each input query, the output which best verbalizes this input. Each output realization associates an input KB query with a NL verbalization and with its TAG derivation trees. From this, we extract for each KB symbol in the input query the TAG tree and the syntactic class used to produce this verbalization.

The resulting training corpus consists of 206 $\langle S, L \rangle$ pairs where S is a linearised KB query and L is the sequence of syntactic labels (TAG tree or syntactic class) associated with each of the KB symbols occurring in S. We learn the hypertagging model on this training corpus[9] using 10-fold cross validation.

4.3.2 Features. All features are derived from the input semantics, that is, a sequence of relations and concepts. Since concepts have low syntactic ambiguity (they mostly select NP trees), most of the features are associated with relations only and in the following, we write R_{i-1} (R_{i+1}) to denote the relation which precedes (follows) relation R_i. Features falls into five major groups: (i) L-Class features, that is, features derived from the shape of relation names which indicate how the relation will be lexicalized and indirectly which TAG tree will be used to verbalize it, (ii) lexical features derived from the words contained in the relation and concept names; (iii) discourse level features indicating how entities relate to each other, that is, whether an entity is common to several relations or whether a new entity is being introduced (topic change); (iv) global structural features pertaining to the overall shape of the input; and (v) combinations thereof.

L-Class Features. We use the lexicalization classes introduced in (Trevisan 2010) as features which provide an abstract characterization of the lexicalization pattern of a KB relation. Each relation in the input is associated with its L-class and with the L-class of the previous and the following two relations. We also use a more general feature describing the semantic type of each relation namely whether it is a binary, a unary or a "compatible unary relation", that is, a concept that labels a query tree node together with other concepts hence, a concept that is compatible with these other concepts.

Lexical features. These features describe characteristics of the words in the concept and relation names, namely whether relations R_{i-1} and R_i have the same names; whether there is a word overlap between the R_i relation name and the following concept name; whether the C_i concept name is an adjective or noun; and whether R_i contains a preposition.

Entity chaining features. These features characterize the distribution of discourse entities in the query linearization. We use three binary features to capture cases where R_{i-1} and R_i, R_{i-2} and R_i, R_{i+1} and R_i share the same first argument, one for cases where the second argument of R_{i-1} is the first argument of R_i; and a feature which summarizes entity sharing between R_{i-1} and R_i by indicating whether or not they predicate over some common entity. There is a feature that captures the changes in topic from the last

9 http://talc1.loria.fr/webnlg/stories/quelo-corpus.tar.gz

mention of an entity in a relation R_{i-k} (where $k = 1, \cdots, i-1$) to the current mention in R_i (only entities in the first argument are considered). This last feature is categorical and encoded as `zero`, `1to2`, `3to4`, `5on` where the first value means that R_i corresponds to a first mention of the entity and the others encode the number of distinct entities mentioned between R_{i-k} and R_i. Finally, an additional feature is used to signal whether the entity denoted by the first argument of R_i is a first mention.

Structural features. This set of features aims at capturing the structure of the query tree and overall query characteristics. Three binary features indicate whether the node in the query tree corresponding to R_i's first (second) argument has children and whether the node corresponding to R_i's first argument has compatible concepts. Two features capture length in terms of number of relations. One captures the length of the sequence of predications ranging over the same entity given by R_i's first argument. The other counts the number of relations to the left of R_i. Both features take the following values `short`,`middle`, `large`.

Feature conjunctions. We use three features combining constraints of different types namely, whether R_{i-1} licences a relational noun and the first argument of R_{i-1} and R_i is the same entity, whether the query tree node corresponding to the second argument of R_i has more than 3 sibling nodes to the left or its immediately preceding sibling node has descendants and whether R_i denotes a unary compatible relation and its first argument is a first mention.

5. Evaluation and Results

In this section, we start by evaluating the impact of the hypertagging module in terms of both speed and coverage. We then go on to evaluate the quality of the generator output when compared to both a template- and a grammar-based approach using both quantitative metrics (BLEU) and a human-based evaluation. In Section 6, we will also show that our approach can account for various factors impacting aggregation, sentence segmentation and the choice of contextually appropriate syntactic structures.

5.1 Impact of the Hypertagging Module on Speed and Coverage

We evaluate the hypertagging module both in isolation and in terms of speed and coverage in interaction with the generator.

Table 3
Hypertagger accuracy (percent). n is the number of best sequences considered.

	Trees		Synt.Classes	
n	Tokens	Input	Tokens	Input
1	63.62	32.05	76.53	49.98
5	77.42	50.90	92.06	78.60
10	82.97	57.64	95.84	86.93

5.1.1 Hypertagging Accuracy. The results for hypertagging accuracy are shown in Table 3[10]. Token accuracy indicates the ratio of input literals correctly labeled while Input accuracy indicates the ratio of input sequences correctly labeled. The two hypertaggers handle 70 tree names and 10 syntactic classes as labels respectively. As is to be expected given the difference in the number of classes to be learned, the results clearly show that both in terms of tokens and in terms of whole inputs, hypertagging is more accurate using syntactic classes than trees.

5.1.2 Generator Performance. Table 4 shows how the hypertagger impacts realization performance in terms of coverage and in terms of speed.

Table 4
Generation Coverage (Percentage of input for which generation produced an output) and time (in ms). Time (gen) is the average time for those inputs for which generation succeeds. Averaged Lexical Ambiguity is an indicator of the number of trees passing through the hypertagging filter. FLS backoff allows for full lexical selection in case hypertagging predicts an incorrect class for a given input literal. n is the number of sequences let through by the hypertagger.

		$n=1$	$n=4$	$n=12$
Trees	Lex Ambiguity	1.01	1.21	1.45♠
	Coverage	64.08	81.07	90.78
	Time (gen)	269	533	725
	Lex Ambiguity-FLS	1.52♣	1.39	1.49♣
	Coverage-FLS	91.75	94.17	95.63
	Time-FLS (gen)	711	806	905
Synt.Cl.	Lex Ambiguity	1.45♠	2.18	3.23
	Coverage	91.26	***97.57***	93.20
	Time (gen)	480	1425	3112
	Lex Ambiguity-FLS	1.49♣	2.17	3.23
	Coverage-FLS	94.66	***98.06***	93.20
	Time-FLS (gen)	520	1414	3113
Symb	Coverage 51.46, avg time 5940, avg lex. ambiguity 5.66			

Coverage is the ratio of input for which the generator outputs a sentence within a time limit of 30 seconds. We set this time limit relatively high to allow more coverage by the symbolic generator. We evaluate coverage using 10-fold cross validation on the training set[11] and experiment with 80 configurations depending on (i) the type of label used by the hypertagger (Trees vs. Syntactic Classes), (ii) the number of sequences let through by the hypertagger (n = 1 to 20) and (iii) whether Full Lexical Selection (FLS) backoff is used. FLS backoff occurs whenever the labels assigned by the hypertagger to a given input literal are not compatible with those specified by the lexicon. In this case,

10 We did regularization parameter selection for both Trees and Synt.Cl models using 10-fold cross validation. The values in this evaluation were obtained using l_1 with $\alpha = -0.20$ for the first model and l_2 with variance 1.5 for the second.

11 For each fold (containing n input semantics), we train a hypertagging model HT on the other 9 folds, call the generator with hypertagging model HT on the n input semantics and retrieve the number of input semantics for which the generator produced a sentence. The total coverage is the sum of the coverage obtained for each fold.

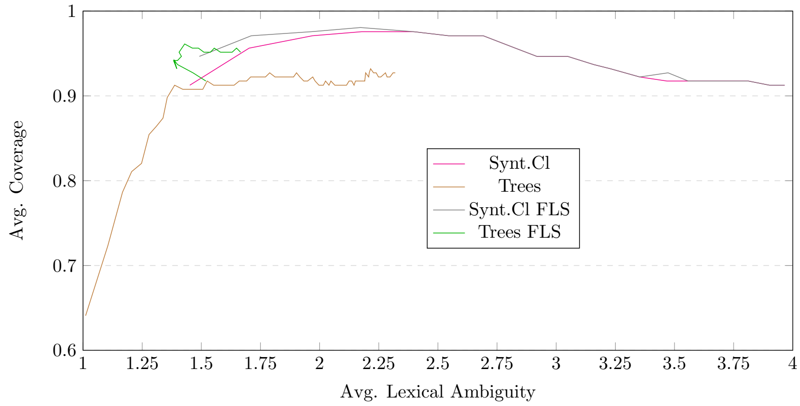


Figure 4
Coverage with respect to avg. lexical ambiguity. Data points are obtained from the different configurations, with/without FLS and n-best sequences with $n = 1..20$; except for the Trees without FLS where the results include up to $n = 70$ configuration.

the hypertagger prediction is ignored and all grammar trees assigned to that literal by the lexicon are selected and considered for tree combination.

Because the syntactic classes used in the Synt.Class configuration describe sets of trees (cf. Section 3.3.4), the number of trees let through by each n-best sequence will be higher for the Syntactic Classes- than for the Tree-Based hypertagger. We therefore indicate coverage and time results not only for different values of n but also in relation to the level of lexical ambiguity allowed by each configuration. Given an input p of length k with literals $l_1, \ldots, l_k$ and t_i the number of selected trees for the literal l_i, we define $LA(p)$, the Lexical Ambiguity of p, as

$$LA(p) = \frac{\sum_{i=1}^{k} t_i}{k}$$

The average lexical ambiguity of m inputs is then

$$\frac{\sum_{j=1}^{m} LA(p_j)}{m}$$

In Table 4, we report results for the 1, 4 and 12 -best sequences which show coverage and time results for the hypertaggers at a comparable degree of lexical ambiguity as well as the maximum coverage achieved. Without the FLS backoff mechanism and at comparable lexical ambiguity of 1.45 (n=1 for the Synt.Cl and n=12 for the Tree hypertagger) the hypertagger with syntactic classes gets slightly better coverage and generation times than the Tree based hypertagger. When using FLS, comparable lexical ambiguity is at n=1 for Synt.Cl with the Tree hypertagger at n=1 and n=12. Within the 1-best sequence configurations the increased lexical ambiguity comes from the FLS backoff; however, in the 12-best sequence configuration more ambiguity comes from the syntactic labels at each sequence and less from the FLS mechanism. Thus at n=1 coverage and time are better for syntactic classes but when looking at n=12 for trees we can see slightly better results. The maximum coverage is achieved by the Synt.Cl hypertagger at n=4 both with and without the FLS backoff. There is a marked difference in coverage (+46.6% w.r.t the Synt.Cl n=4) between the hypertaggers and the symbolic generator which often times out on the unrestricted search space.

For a complete picture of the results with all configurations, in Figure 4 we draw coverage w.r.t lexical ambiguity at each n-best configuration. We run the Tree based hypertagger up to the 70-best sequences configuration with no FLS to find out which is the maximum coverage that could be attained. In all configurations from $n = 21, \cdots, 70$, the Tree tagger oscillates in coverage between 91.75 and 92.72 while average lexical ambiguity and time increase at each $(n + 1)$ -best configuration (coverage 92.72 avg. lex.ambiguity 2.32, avg. time 1820 at n=70).

In sum, hypertagging using syntactic classes permits improving efficiency by a very wide margin with respect to the symbolic generator (1414 ms/input vs. 5940 ms/input) while preserving coverage (98.06% vs. 51.46%).

5.2 Quality of the Generated Texts

Input Query	`Flight` $\sqcap$ $\exists$`hasCurrentDepartureDate.Date` $\sqcap$ $\exists$`hasCurrentArrivalDate.Date` $\sqcap$ $\exists$`hasDestination.Airport hasFlightTo.Airport` $\sqcap$ $\exists$`hasCarrier.Airline` $\sqcap$ $\exists$`hasTicket.AirTicket` $\sqcap$ $\exists$`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.

Figure 5
Example input and outputs. Temp is a template based system, Symb the symbolic generator described in Section 3.3.3 and Hyb the same generator augmented with the Hypertagger.

The quality of the generated natural language queries is evaluated using a subject human rating study which aims to determine whether the queries generated by our hybrid generation system (Hyb[12]) are perceived as better by human judges than those generated for the same inputs by a template based system (Temp) and by a grammar-based generator without the hypertagging module (Symb). Figure 5 shows an example input and the output produced by each system.

The template-based system is a generation system previously developed for the Quelo natural language interface which uses templates to verbalize binary relations and their arguments. This template-based version of Quelo generates one clause per relation, post-processes referring expressions and allows for some forms of aggregation. For instance, two subject sharing relations may be realized in the same clause. Example (9a) shows an example output produced by the template-based version of Quelo.

12 The configuration we use in this evaluation is the Synt.Cl hypertagger with 4-best and FLS.

(9) 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.

The grammar-based system is our system without the hypertagging module.

For each generation system, we consider a single output. The template system is deterministic and always returns a single output. For the symbolic approach, we use a symbolic quality score provided by the system while for the hybrid system, we use the best scored sentence generated when using the class-based hybertagger with full lexical selection backoff and 4-best only.

The evaluation was done using the Crowdflower platform[13]. In this evaluation, contributors were shown two verbalizations of the same input but produced by two different systems and asked to score those two systems on a scale of 1 to 3 in terms of fluency (How well does the sentence read? Is the text well structured?) and clarity (How easy is it to understand?).

We collected ratings for sentences generated by each of the three systems from 49 input queries built from 13 knowledge bases describing different domains. Five of the input queries were built from two knowledge bases (Air System and Bibliography Ontologies) that were not present in the training corpus. Each generated sentence was rated by at least 10 contributors.

Crowdflower implements a quality control system based on test sentences which have a pre-determined gold-standard answer and are indistinguishable from other sentences. We used a set of 10 test questions. In order to participate, contributors had to pass a "Quiz Mode" consisting of test sentences for which they needed to obtain a minimum accuracy of 60%. They then had to maintain this accuracy throughout the job.

Table 5 shows the aggregated results of the CrowdFlower evaluation. The aggregate rating of a sentence is chosen based on the following confidence score:

$$conf(a \mid q) = \frac{\sum_{c \in C_a} acc(c)}{\sum_{c \in C_a} acc(c)}$$

where C_a is set of contributors who responded to question q with answer a, C_a the total set of contributors who responded to question q and $acc(c)$ is the accuracy of contributor c. The average confidence of the data is 0.67% for the fluency evaluation and 0.68% for clarity.

Overall, the hybrid system yields output that is consistently perceived by the human raters as clearer and more fluent.

67% of the texts generated by the template based system are rated as non fluent (against 2% for the hybrid approach) and only 47% of these texts are rated as clear (against 96% for the hybrid approach). We conjecture that the low fluency is related to the lack of structuring elements (often the template system yields one sentence per binary relation thereby producing text that is a juxtaposition of short sentences). Concerning clarity, we believe that the repetitions resulting from the restricted aggregations allowed by the template system make it difficult to detect the links between multiple descriptions of the same entity and indirectly, to understand the meaning of the generated text.

13 `http://www.crowdflower.com`

Table 5
Output Quality.

Criteria		Symb/Hyb				Temp/Hyb			
		Symb		Hyb		Temp		Hyb	
		Ratio	Nb	Ratio	Nb	Ratio	Nb	Ratio	Nb
Fluency	Fluent	8%	4	65%	32	2%	1	51%	25
	Medium	43%	21	29%	14	31%	15	47%	23
	Non Fluent	49%	24	6%	3	67%	33	2%	1
Clarity	Clear	8%	4	59%	29	47%	23	96%	47
	Medium	90%	44	41%	20	53%	26	4%	2
	Unclear	1%	1	0%	0	0%	0	0%	0

Although the gap between the symbolic and the hybrid approach is less marked than between the template and the hybrid system, the purely symbolic system also scores less well than the hybrid approach which may be due to the fact that the symbolic generator often fails to adequately segment the input or to score the most fluent output highest (only 8% of the text generated by the symbolic system are rated as clear by the annotator against 59% for the hybrid approach).

Table 6
BLEU Scores

System	BLEU/all	BLEU/gen
Temp	0.59	0.59
Symb	0.37	0.72
Hyb Synt.Cl. FLS n=1	0.80	0.85
Hyb Synt.Cl. FLS n=4	0.73	0.75
Hyb Trees FLS n=1	0.78	0.84
Hyb Trees FLS n=10	0.76	0.79

We also evaluate system output automatically, using the BLEU-4 modified precision score (Papineni et al. 2002) on the gold query verbalizations in the training corpus and in a 10 fold cross validation setting as explained in the previous section (Section 5.1). We computed BLEU scores for all inputs (BLEU/all) and for those 206 inputs for which the generators yielded an output (BLEU/gen). The results given in Table 6 show that our hybrid system produces query verbalizations that are closer to the manually selected gold query verbalizations than both the template and the purely symbolic grammar-based approach. Although the BLEU/gen score for the Symb system is relatively high, it decreases drastically when normalized by coverage because of time outs on long inputs.

6. Interactions between Segmentation, Aggregation and Surface Realization

In this section, we illustrate by means of examples how our approach accounts for various factors impacting sentence segmentation, aggregation and surface realization.

6.1 Sentence Segmentation

The shape and the size of the input data influences the segmentation of this data into clauses and sentences. The input/output pairs in example (10) illustrate how our generation approach yields different segmentations for inputs which differ in terms of structure and length.

Examples (10a-b) show two inputs including respectively, 3 and 4 relations. While the hypertagger predicts a single sentence for the shorter input (10a), it correctly accounts for the additional length in (10b) by predicting a segmentation into 2 sentences. Similarly, (10c) differs from (10a) in that it includes one more concept. Again this induces differences in segmentation that are consistent with intuition.

(10) a. `UsedCar ⊓ ∃exteriorColor.White ⊓ ∃locatedInCountry.Country ⊓ ∃hasModel.Toyota4Runner` *(3 relations, 4 concepts, 1 sentence)*
I am looking for a used car whose exterior color should be white and which should be located in a country and whose model should be a toyota 4 runner.
b. `NewCar ⊓ ∃ exteriorColor.Beige ⊓ ∃hasCarBody.UtilityVehicle ⊓ ∃runOn.NaturalGas ⊓ ∃locatedInCountry.Country` *(4 relations, 5 concepts, 2 sentences)*
I am looking for a new car whose exterior color should be beige and whose body style should be a utility vehicle. The new car should run on a natural gas and should be located in a country.
c. `NewCar ⊓ ∃hasCarBody.(UtilityVehicle ⊓ OffRoad) ⊓ ∃runOn.NaturalGas ⊓ ∃locatedInCountry.Country` *(3 relations, 5 concepts, 2 sentences)*
I am looking for a new car whose body style should be a utility vehicle, an off road. The new car should run on a natural gas and should be located in a country.

6.2 Surface Realization

A given lexicalization may give rise to different syntactic realizations depending on the context. The following examples show that our approach can choose different syntactic constructions for one and the same relation. Depending on the shape and content of the input formula, the `teach` relation is verbalized in four different ways (relative clause, coordinated relative clause, coordinated VP, canonical clause).

(11) a. `TeachingAssistant ⊓ ∃teach.Course ⊓ ∃employedBy.University`
I am looking for a teaching assistant ***who should teach a course*** *and should be employed by a university.* *(Relative Clause)*
b. `Professor ⊓ Researcher ⊓ ∃teach.Course`
I am looking for a professor who is a researcher ***and who should teach a course****.* *(Coordinated Relative Clause)*
c. `Professor ⊓ ∃isCoordinatorOf.MastersProgram ⊓ ∃supervise.MastersThesis ⊓ ∃teach.Course`
I am looking for a professor who should be the coordinator of a masters program, should supervise a masters thesis ***and should teach a course****.* *(Coordinated VP)*
d. `MastersProgram ⊓ hasCoordinator.(Coordinator ⊓ Researcher ⊓ ∃teach.Course ⊓ ∃employedBy.University)`
I am looking for a masters program whose coordinator should be a coordinator, a researcher. The coordinator ***should teach a course*** *and should be employed by a university.* *(Canonical Clause)*

Similarly, Example (12) shows four distinct surface realizations produced by our generator for the `locatedin` relation.

(12) a. `CarDealer ⊓ ∃locatedInCountry.Country ⊓ ∃sell.(Car ⊓ ∃hasMake.Toyota ⊓ ∃runOn.Fuel ⊓ ∃equippedWith.ManualGearTransmission)`
*I am looking for a car dealer **located in a country** and who should sell a car whose make should be a toyota. The car should run on a fuel and should be equipped with a manual gear transmission system.* *(Participial)*
b. `CarDealer ⊓ ∃sell.(NewCar ⊓ ∃hasModel.Toyota ⊓ ∃locatedInCountry.Country)`
*I am looking for a car dealer who should sell a new car whose model should be a toyota. **It should be located in a country**.* *(Canonical Clause with pronominal subject)*
c. `NewCar ⊓ OffRoad ⊓ ∃hasCarBody.UtilityVehicle ⊓ ∃runOn.NaturalGas ⊓ ∃locatedInCountry.Country`
*I am looking for a new car, an off road whose body style should be a utility vehicle. The new car should run on a natural gas **and should be located** in a country.* *(Coordinated VP)*
d. `Car ⊓ ∃producedBy.(CarMake ⊓ ∃isMakeOf.Toyota ⊓ ∃locatedIn.City ⊓ ∃produceModel.LandRoverFreelander`
*I am looking for a car produced by a car make. The car make should be the make of a toyota. The car make **should be located** in a city and should produce a land rover freelander.* *(Canonical Clause)*

6.3 Aggregation

The syntactic variability encoded in our grammar and the choices made by the hypertagging module account for various cases of aggregation.

Relative clauses permit linking multiple propositions within a single sentence. When more than two propositions are present, the grammar permits distinguishing between cases where multiple predications apply to the same concept (Example 13a) and cases where predications are chained (Example 13b). Thus the pattern `C1 (R1 C2) (R2 C3)` (Example 13a) will be verbalized as *N1 WH V1 N2 and WH-V2 N3* while the pattern `C1 (R1 C2 R2 C3)` (Example 13b) will be verbalized as *N1 WH V1 N2 WH-V2 N3*.

(13) a. `Concert ⊓ ∃hasDestination.Destination ⊓ ∃hasSite.Site`
I am looking for a concert whose destination should be a destination and whose site should be a site *(X which VP1 and which VP2)*
b. `CarDealer ⊓ ∃sell.(NewCar ⊓ ∃hasModel.Toyota)`
I am looking for a car dealer who should sell a new car whose model should be a toyota. *(X which R1 Y whose R2 should be Z)*

Cases involving more than two propositions can also be accounted for whereby pied piping and participial constructions can interact with relative clauses to produce a complex sentence out of several propositions.

(14) a. `Toyota ⊓ ∃isMakeOf.(NewCar ⊓ ∃runOn.Gas) ⊓ ∃isMakeOfmodel.LandRoverDefender`
I am looking for a toyota <u>which</u> should be the make of a new car <u>which</u> should run on a gas. The Toyota should be the make of a land rover defender.
b. `CarDealer ⊓ ∃locatedInCountry.Country ⊓ ∃sell.(Car ⊓ ∃hasMake.Toyota ⊓ ∃runOn.Fuel)`
I am looking for a car dealer <u>located</u> in a country <u>and who</u> should sell a car <u>whose make</u> should be a toyota. The car should run on a fuel.
c. `Movie ⊓ ∃producedBy.Producer ⊓ ∃writtenBy.Writer ⊓ ∃hasGenre.Genre`
I am looking for a movie <u>produced</u> by a producer, <u>written</u> by a writer and <u>whose genre</u> should be a genre

The grammar also allows for verbal ellipsis using VP-, relative clause- and NP-coordination (Examples 15).

(15) a. `NewCar ⊓ ∃exteriorColor.Beige ⊓ ∃hasCarBody.UtilityVehicle ⊓ ∃runOn.NaturalGas ⊓ ∃locatedInCountry.Country`
I am looking for a new car whose exterior color should be beige and whose body style should be a utility vehicle. The new car (should run on a natural gas and should be located in a country)$_{VP}$.
b. `CommunicationDevice ⊓ ∃assistsWith.Understanding ⊓ ∃assistsWith.HearingDisability`
I am looking for a communication device (which should assist with a understanding and which should assist with a hearing disability)$_{RelCl}$.
c. `CarDealer ⊓ ∃sell.CrashCar ⊓ ∃sell.NewCar`
I am looking for a car dealer who should sell (a crash car and a new car)$_{NP}$.
d. `Car ⊓ ∃equippedWith.ManualGearTransmission ⊓ ∃equippedWith.AlarmSystem ⊓ ∃equippedWith.NavigationSystem ⊓ ∃equippedWith.AirBagSystem`
I am looking for a car equipped with (a manual gear transmission system, an alarm system, a navigation system and an air bag system)$_{NP}$.

When a new sentence is started, referring expressions may be realized either by full NPs (Example 16a) or by pronouns (Example 16b).

(16) a. `CarMake ⊓ ∃locatedInCountry.Country ⊓ ∃isMakeOfModel.LandRoverDiscovery ⊓ ∃isMakeOf.DemonstrationCar`
I am looking for a car make located in a country. The car make should be the make of a land rover discovery and should be the make of a demonstration car.
b. `Car ⊓ ∃producedby.CarMake ⊓ ∃soldBy.(CarDealer ⊓ ∃locatedInCountry.Country ⊓ ∃equippedWith.NavigationSystem ⊓ ∃equippedWith.Abs ⊓ ∃equippedWith.GasolineEngine)`
I am looking for a car produced by a car make and sold by a car dealer located in a country. It should be equipped with a navigation system, an abs and a gasoline engine.

7. Conclusion

Recent statistical approaches to NLG (Konstas and Lapata 2012b, 2012a; Wong and Mooney 2007; Angeli, Liang, and Klein 2010) have typically relied on sizable training corpora to train models that directly map meaning representations to strings. In contrast, we developed a hybrid model which integrates a statistical hypertagger in a grammar based generation system. This has several advantages.

First, because the grammar used is generic and the hypertagger trained to learn syntactic units (rather than e.g., domain dependent semantic ones), the approach is domain independent and can be applied to any knowledge base. More specifically, applying the approach to a new KB requires neither developing a new parallel data-to-text corpus for training nor developing a new grammar.

Second and most importantly, using a grammar permits modeling sub-sentence level phenomena such as sentence- and VP-coordination, relative clauses and ellipsis. This is in stark contrast with the template-based approaches often used in data-to-text generation where aggregation and surface realization choices are either hardcoded in sentential templates (Kondadadi, Howald, and Schilder 2013; Duma and Klein 2013) or enforced by ad-hoc aggregation rules introduced to combine two or more sentence templates into a complex sentence. In comparison, our grammar-based approach nat-

14 http://linkeddata.org/home

Trees	Synt.Cl
betavx0ANDVVDNpnx1, betavx0ANDVVnx1, betavx0ANDVVVpnx1, betavx0ANDVVpnx1, betavx0COMMAVVnx1, betavx0COMMAVVVpnx1, betanx0ANDVPnx1, betanx0COMMAVPnx1, betanx0COMMAVApnx1, betanx0ANDPnx1, betanx0COMMAPnx1, betanx0COMMAVPpnx1, betanx0ANDVPpnx1	SharedSubj
betavx0COMMAnx1, betavx0CONJnx1	Ellipsis
sDOTDNPnx0VVax1, sDOTDNPnx0VVnx1, nx0BEnx1, sDOTnx0VVDNpnx1, sDOTnx0VVnx1, sDOTnx0VVpnx1, sDOTnx0VVVnx1, sDOTnx0VVVpnx1, sDOTnx0VVApnx1, sDOTPRO0VVnx1, sDOTPRO0VVVpnx1, sDOTPRO1NVVnx1, sDOTPRO1NVVax1, sDOTPRO0VVDNpnx1, sDOTPRO0VVVnx1	Canonical
betanx0VPpnx1, betanx0Pnx1	PpartOrGerundOrPrerp
betanxBEnx, Npx0VVax1, Npx0VVnx1, W0nx0VVApnx1, W0nx0VVDNpnx1, W0nx0VVnx1, W0nx0VVpnx1, W0nx0VVVnx1, W0nx0VVVpnx1, W0nx0VVNpnx1	SubjRel
ANDNpx0VVax1, ANDNpx0VVnx1, ANDWHnx0VVDNpnx1, ANDWHnx0VVnx1, ANDWHnx0VVpnx1, ANDWHnx0VVVnx1, ANDWHnx0VVVpnx1, ANDWHnxBEnx, COMMAWHnx0VVnx1, COMMAWHnx0VVVpnx1, COMMAWHnx0VVDNpnx1, COMMAWHnx0VVVpnx1, COMMANpx0VVnx1	SubjRelPU
sCONJnx0VVpnx1, sCONJnx0VVApnx1, sCONJnx0VVVpnx1, sCONJDNPnx0VVnx1, sCONJPRO0VVVpnx1, sCONJPRO1NVVax1, sCONJPRO1NVVnx1, sCONJPRO0VVnx1, sCONJPRO0VVpnx1, sCONJPRO0VVDNpnx1, sCONJPRO0VVApnx1	S-Coordination
betanxPUnx	Apposition
ax, nx	Unary-argument
betaADJnx	Word-adjunct

Table A1
List of TAG trees and syntactic classes.

urally supports aggregation by allowing for the generation of relative clauses, gerund, elliptical and coordinated constructions.

Third, the hybridation of a symbolic grammar based surface realizer with a statistical hypertagger permits capturing both hard grammaticality constraints and the softer acceptability constraints regulating the interplay between grammatical structures, linearization, lexicalization and topic structure.

There are many possible directions for further research.

One first issue is how to improve lexicalization. Because lexicalization was not our focus here, we assumed a simple lexicalization procedure based on the shape of the relation names. As the examples in the previous section clearly show, this is not always appropriate. It would be interesting to explore ways of going beyond such a simple procedure.

Another interesting issue concerns generation from linked data[14] and RDF triples. Linked data and RDF triples provide a generic graph-based data model for describing things and their relationships with other things. They are in this sense very similar to the binary relations and assertions contained in Knowledge Bases. We are currently exploring both whether and if so, how the grammar we use could be extended to cover text generated from such data; as well as whether the sentence segmentation enforced by the hypertagger generalizes to the segmentation of text generated from linked data.

Finally, we are interested in exploring how our approach could be applied to more complex text and more complex data and in particular, how it could be extended to handle the interactions between discourse connectives and micro-planning. Indeed the approach we have described here is currently restricted to simple data (conjunctive tree shape queries) and relatively simple text (limited discourse structure). In the future, we intend to investigate whether a similar hybrid approach could be used to generate structured discourse from more complex data such as for instance, Abstract Meaning Representations (AMR, (Banarescu et al. 2012)), representations produced by machine reading tools such as Fred (Draicchio et al. 2013) or the Discourse Representations Structures derived by Boxer (Bos 2008).

Appendix A: List of TAG trees and Syntactic Classes Used by the Hypertagger

Table A1 shows the list of TAG trees and abstract syntactic classes used in the annotation of the training corpus. The tree names follow the naming conventions of TAG. Upper case letters indicate anchors. Phrasal projections are postfixed with the "x" symbol and integers indicate semantic role (e.g., 0 for for the first argument of a relation, 1 for the second). A beta prefix indicates an auxiliary tree. The W prefix indicates a wh-NP extraction. Thus for instance, W0nx0VVVnx1 indicates a subject relative tree (W0nx0) with three verbs as anchor (VVV) and an NP as complement (nx1).

Acknowledgments

We thank the ANR and the ITEA 3 for funding the research leading to this article via the WebNLG and the ModelWriter projects. We would also like to thank Marc Dymetman, Elena Lloret, Christophe Cerisara and the members of the SEPLN conference, of the XRCE seminar in Grenoble, of the ILCC/HCRC seminar in Edinburgh and of the Brown Bag Seminar at the National Library of Medicine in Bethesda for valuable feedback and discussions.

References

Angeli, Gabor, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language*

Processing, pages 502–512. Association for Computational Linguistics.
Appelt, Douglas E. 1982. Planning natural-language utterances to satisfy multiple goals. Technical report, DTIC Document.
Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2012. Abstract meaning representation (amr) 1.0 specification. In *Parsing on Freebase from Question-Answer Pairs.âĂİ In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL*, pages 1533–1544.
Bangalore, Srinivas and Aravind K Joshi. 1999. Supertagging: An approach to almost parsing. *Computational linguistics*, 25(2):237–265.
Bangalore, Srinivas and Owen Rambow. 2000. Using TAG, a tree model, and a language model for generation. In *In Proceedings of the 1st International Natural Language Generation Conference*. Citeseer.
Bos, Johan. 2008. Wide-coverage semantic analysis with boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pages 277–286. Association for Computational Linguistics.
Clark, Stephen and James R Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th international conference on Computational Linguistics*, page 282. Association for Computational Linguistics.
Crabbé, Benoît, Denys Duchier, Claire Gardent, Joseph Le Roux, and Yannick Parmentier. 2013. eXtensible Meta Grammar. *Computational Linguistics*, pages 581–620.
Curran, James R, Stephen Clark, and David Vadas. 2006. Multi-tagging for lexicalized-grammar parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 697–704. Association for Computational Linguistics.
Danlos, Laurence. 1987. The linguistic basis of text generation. In *Proceedings of the third conference on European chapter of the Association for Computational Linguistics*, pages 1–1. Association for Computational Linguistics.
Dethlefs, Nina, Helen Hastie, Heriberto Cuayáhuitl, and Oliver Lemon. 2013. Conditional random fields for responsive surface realisation using global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1254–1263, Sofia, Bulgaria, August. Association for Computational Linguistics.
Dongilli, Paolo. 2008. Natural language rendering of a conjunctive query. *KRDB Research Centre Technical Report No. KRDB08-3). Bozen, IT: Free University of Bozen-Bolzano*, 2:5.
Draicchio, Francesco, Aldo Gangemi, Valentina Presutti, and Andrea Giovanni Nuzzolese. 2013. Fred: From natural language text to rdf and owl in one click. In *The Semantic Web: ESWC 2013 Satellite Events*. Springer, pages 263–267.
Duma, Daniel and Ewan Klein. 2013. Generating natural language from linked data: Unsupervised template extraction. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 83–94, Potsdam, Germany, March. Association for Computational Linguistics.
Espinosa, Dominic, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of ACL-08: HLT*, pages 183–191, Columbus, Ohio, June. Association for Computational Linguistics.
Ferrandez, Oscar, Christian Spurk, Milen Kouylekov, Iustin Dornescu, Sergio Ferrandez, Matteo Negri, Ruben Izquierdo, David Tomas, Constantin Orasan, Guenter Neumann, et al. 2011. The QALL-ME framework: A specifiable-domain multilingual question answering architecture. *Web semantics: Science, services and agents on the world wide web*, 9(2):137–145.
Franconi, Enrico, Paolo Guagliardo, Sergio Tessaris, and Marco Trevisan. 2011. A natural language ontology-driven query interface. In *9th International Conference on Terminology and Artificial Intelligence*, page 43.
Franconi, Enrico, Paolo Guagliardo, and Marco Trevisan. 2010a. An intelligent query interface based on ontology navigation. In *Proceedings of the Workshop on Visual Interfaces to the Social and Semantic Web (VISSW 2010)*, volume 565. Citeseer.
Franconi, Enrico, Paolo Guagliardo, and Marco Trevisan. 2010b. Quelo: a NL-based intelligent query interface. In *Proc. of the 2nd Workshop on Controlled Natural Languages (CNL 2010)*.
Franconi, Enrico, Paolo Guagliardo, Marco Trevisan, and Sergio Tessaris. 2011. Quelo:

an Ontology-Driven Query Interface. In *Proceedings of the 24th International Workshop on Description Logics (DL 2011), Barcelona, Spain, July 13-16, 2011*.

Gardent, Claire and Eric Kow. 2007. A symbolic approach to Near-Deterministic Surface Realisation using Tree Adjoining Grammar. In *ACL*, volume 7, pages 328–335.

Gardent, Claire and Laura Perez-Beltrachini. 2010. RTG based surface realisation for TAG. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, pages 367–375, Beijing, China, August. Coling 2010 Organizing Committee.

Guagliardo, Paolo. 2009. Theoretical foundations of an ontology-based visual tool for query formulation support. Technical report, KRDB Research Centre, Free University of Bozen-Bolzano, October.

Keet, C Maria, Ronell Alberts, Aurona Gerber, and Gibson Chimamiwa. 2008. Enhancing Web Portals with Ontology-Based Data Access: The Case Study of South Africa's Accessibility Portal for People with Disabilities. In *OWLED*, volume 432.

Kondadadi, Ravi, Blake Howald, and Frank Schilder. 2013. A statistical nlg framework for aggregated planning and realization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1406–1415, Sofia, Bulgaria, August. Association for Computational Linguistics.

Konstas, Ioannis and Mirella Lapata. 2012a. Concept-to-text generation via discriminative reranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 369–378. Association for Computational Linguistics.

Konstas, Ioannis and Mirella Lapata. 2012b. 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*, pages 752–761. Association for Computational Linguistics.

Lafferty, John, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lampouras, Gerasimos and Ion Androutsopoulos. 2013. Using integer linear programming in concept-to-text generation to produce more compact texts. In *51st Annual Meeting of ACL (short papers)*, pages 561–566.

Lu, Wei, Hwee Tou Ng, and Wee Sun Lee. 2009. 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*, pages 400–409. Association for Computational Linguistics.

Marciniak, Tomasz and Michael Strube. 2005. Beyond the pipeline: Discrete optimization in NLP. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 136–143. Association for Computational Linguistics.

McCallum, Andrew Kachites. 2002. Mallet: A machine learning for language toolkit.

Mel'čuk, IgorÊź Aleksandrovič. 1988. *Dependency syntax: theory and practice*. SUNY Press.

Meteer, Marie Wenzel. 1990. The"generation gap": The problem of expressibility in text planning.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Perez-Beltrachini, Laura, Claire Gardent, and Enrico Franconi. 2014. Incremental query generation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 183–191, Gothenburg, Sweden, April. Association for Computational Linguistics.

Reiter, Ehud, Robert Dale, and Zhiwei Feng. 2000. *Building natural language generation systems*, volume 33. MIT Press.

Trevisan, Marco. 2010. 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).

Vijay-Shanker, K and Aravind K Joshi. 1988. Feature structures based tree adjoining grammars. In *Proceedings of the 12th conference on Computational linguistics-Volume 2*, pages 714–719. Association for Computational Linguistics.

Walker, Marilyn A, Owen Rambow, and Monica Rogati. 2001. Spot: A trainable sentence planner. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational*

Linguistics on Language technologies, pages 1–8. Association for Computational Linguistics.

Walker, Marilyn A, Amanda Stent, François Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research*, pages 413–456.

Wong, Yuk Wah and Raymond J Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *HLT-NAACL*, pages 172–179.

Zarrieß, Sina and Jonas Kuhn. 2013. Combining Referring Expression Generation and Surface Realization: A Corpus-Based Investigation of Architectures. In *Proc. of ACL*, pages 1547–1557.