Quelo Natural Language Interface: Generating queries and answer descriptions

Enrico Franconi[‡], Claire Gardent[*], Ximena I. Juarez-Castro[‡], and Laura Perez-Beltrachini[‡*]

[‡]Faculty of Computer Science, Free University of Bozen-Bolzano
Piazza Domenicani, 3, Bozen-Bolzano BZ, Italy
[*]CNRS/LORIA
Campus Scientifique, BP 239, Vandoeuvre-lès-Nancy, France
{claire.gardent,laura.perez}@loria.fr
franconi@inf.unibz.it
Ximena.JuarezCastro@stud-inf.unibz.it

Abstract. We present an intelligent NLI interface, namely Quelo NLI, for querying and exploring semantic data. Its intelligence lies in the use of reasoning services over an ontology. These support the intentional navigation of the underlying datasource and the formulation of queries that are consistent with respect to it. Its Natural Language Generation (NLG) module masks the formulation of queries as the composition of English text and generates descriptions of query answers. An important feature of Quelo NLI is that it is portable as it is not bound to an ontology of a specific domain. We describe Quelo NLI functionality and present a grammar-based natural language generation approach that better supports the domain-independent generation of fluent queries and naturally extends for the generation of answers descriptions. We concentrate on describing the generation resources, namely a domain-independent handwritten grammar and a lexicon that is automatically extracted from concepts and relations of the underlying ontology.

Keywords: ontology-based data access, intentional navigation, conceptual authoring, natural language generation

1 Introduction

Natural Language Interfaces (NLI) to access structured data have been proposed and studied since many years now. Early research dates back to pioneer work by [34,35] (cf. [1] for a survey) and more recent work includes [29,5,23,17], among others. A major motivation behind all of this work is to relieve casual-users from the necessity of learning formal languages, of knowing the underlying data source structure and content, and of learning the functioning of graphical interfaces (cf. [20]). A major goal of this research in NLIs is thus to equip systems with useful Natural Language (NL) based communication capabilities. For instance, some systems (e.g. TEXT [35]) answer questions about the structure of the database;

while others (e.g. WebCoop [2]) rely on mechanisms to detect erroneous assumptions in the users request and try to sort out the situation by engaging in some clarification process.

In this paper we present an extended NLI for the Quelo Tool[1] [7,8,9], namely Quelo NLI, which supports users in formulating a precise query over an ontology by means of an intelligent interface using natural language. Query formulation is an interactive and incremental process. Quelo Tool's query manipulation operations compute relevant query revisions with respect to the current query and the underlying Knowledge Base (KB). These revisions are suggested to the user who will choose amongst them. In this way, users are guided in the formulation of their request. They do not need to know in advance the organisation of the data nor the vocabulary of the modelled domain; the suggested revisions expose the terms used to name things thereby the user will explore, acquire and use the terminology employed by the system (cf. [10]). An important consequence of the Quelo Tool guidance is that it prevents the user from formulating misconceived queries and going through the potentially frustrating and time consuming process of finding out which is the right question to ask.

In Quelo NLI, the query built so far is represented by one or more natural language sentences and query revisions are suggested to the user in natural language. Following the "conceptual authoring" approach described in [30,16], Quelo NLI masks the composition of a formal query as the composition of an English text describing the equivalent information needs using Natural Language Generation (NLG) techniques. The user starts from a sentence expressing a general request and will refine it by choosing the convenient details for their request from the suggested NL revisions.

Quelo NLI also extends the use of NL to the presentation of the answer after query execution. When a query is executed, a description of the resulting tuples with respect to the user's NL query is automatically generated. This description will potentially help the user to relate the query with the obtained set of tuples, to interpret each resulting tuple by describing the relation between its component elements and to verify whether what they obtained as result is what they intended to get.

NLG from KBs has been used to provide NLIs within two main application contexts ([13]): the generation of reports and summaries ([3,11,4]) and the verbalisation of KBs for engineering purposes such as verbalising ontologies to document knowledge bases or to facilitate communication between knowledge experts and users ([19,26,22]). In the former case, sophisticated NLG components relying on domain dependent resources (e.g. manually authored lexicons or domain-dependent corpora) are developed to produce high quality text. In contrast, in the later case, domain-independent generators assuming a one-to-one mapping from data to text are used to produce verbalisations that are as unambiguous as possible and stay close to the underlying formal language. Indeed, this second type of generators follow the "consensus model" [27] which assumes that atomic concepts and relations are mapped to words and each on-

[1] krdbapp.inf.unibz.it:8080/quelo

tology axiom is mapped to a separate sentence. They are domain-independent in that they produce verbalisations on the fly for any given KB independently of its vocabulary.

Quelo NLI relies on a domain-independent ontology-based query framework ([15]) and thus is conceived to be portable to different domains. In addition, queries will be represented by one or more sentences and query answer descriptions will describe entities of the query from a different perspective. Thus, departing from existing work in generation from KBs and from similar work in query verbalisation ([16,8]), we propose a portable grammar-based approach that supports through a one-to-many data to text mapping the generation of fluent query verbalisations and answer descriptions.

The paper is organised as follows. Section 2 describes Quelo NLI. In Section 3, we describe the generation task and present the overall generation architecture, then we concentrate on the surface realiser and focus on the description of the resources it uses, i.e. grammar and lexicon. We conclude with a discussion in Section 4.

2 Core functionality and user interface

The Quelo Tool ([15]) aims at supporting users in the formulation of a precise query which best captures their information needs against some underlying datasource. In Quelo NLI, these queries are expressed using natural language. More precisely, queries are formulated by adding, replacing or deleting snippets of English text at different points thereof. While users manipulate NL queries, the Quelo Tool operates on formal queries. The NLI interface links text spans of the NL query with elements of the underlying formal query (i.e. concepts and relations). These text snippets prompted to the user are generated automatically using NLG techniques from possible query revisions computed by Quelo on the underlying formal query. Figure 1 shows Quelo NLI, a query that has been formulated so far (i.e. *I am looking for a car.*) and a set of possible query revisions suggested to the user in natural language.

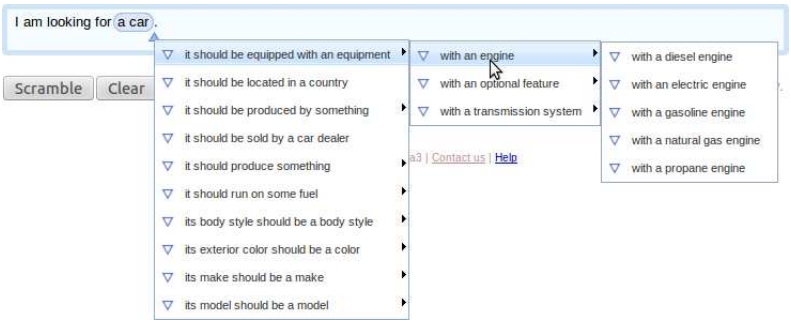

Fig. 1. Quelo's NLI showing a NL query and its possible refinements.

The user interface combines visual elements and click operations for the manipulation of the NL query. Thanks to the link between text and the underlying formal query elements, different visual hints are implemented. For instance, *active text snippets* corresponding to concepts and relations are highlighted (e.g. *car* in Figure 1). When hovering with the mouse on active text snippets, the interface provides visual hints about the structure of the query, by highlighting snippets related with the one on which the hovering occurs. It is also possible to *select* parts of the query by clicking on active text snippets. Clicks on darked-blue triangle-shaped buttons associated with entities in the query (e.g. *car* in Figure 1) bring Quelo suggestions for query refinement.

The query formulation process starts from a general request expressed by the query "I am looking for something". Although the user will have in mind the information they are looking for, at this point they might know nothing about the underlying datasource structure nor its vocabulary. Quelo query manipulation operations let the user explore the underlying datasource and formulate a query that is valid with respect to it. The four operations (cf. [15] for a formal definition of the operations) are: *add* for the addition of new concepts and relations; *substitution* for replacing a portion of the query with a more general, specific or compatible concept; *deletion* for removing a selected part of the query; and *weaken* for making the query as general as possible. A sequence of query formulation steps illustrating these operations is shown in Example (1).

(1) *I am looking for something.* (Initial request)
I am looking for a new car. (Substitution)
I am looking for a new car sold by a car dealer. (Add relation)
I am looking for a new car, a coupé sold by a car dealer. (Add concept)
I am looking for a new car sold by a car dealer. (Deletion)
I am looking for a car sold by a car dealer. (Weaken)

Quelo NLI also provides a query execution option which lets the user see which are those instances satisfying their current request. The query execution can retrieve instances for any of the entities being mentioned in the query. If nothing is marked in the query, e.g. query in Example (2a), the answer to the query will contain instances of the main concept being described in the query, e.g. car. In contrast, if active text snippets corresponding to relations are marked, e.g. ***sold by*** and ***located in*** in Example (2b), instances of the concepts following the relations, e.g. *car dealer* and *city*, are included in the query answer tuples when this is executed.

(2) a. *I am looking for a car sold by a car dealer.*
b. *I am looking for a car* ***sold by*** *a car dealer* ***located in*** *a city.*

Quelo NLI presents the answer set of tuples resulting from query execution in a tabular format with a natural language description for each of the components of the tuple. This description is generated with respect to the current query but from a different perspective as it introduces one or more instances in the tuples answering the query. Figure 2 shows the NL descriptions generated for an example answer set for the query in (2b).

Query results

A car dealer.	The city in which that car dealer is located.
http://www.inf.unibz.it/~dongilli/ontologies/cars4-tiny#Jegla_AG	http://www.inf.unibz.it/~dongilli/ontologies/cars4-tiny#Rickenbach
http://www.inf.unibz.it/~dongilli/ontologies/cars4-tiny#Auto_Center_Wetzikon	http://www.inf.unibz.it/~dongilli/ontologies/cars4-tiny#Wetzikon

Fig. 2. Query answer description.

3 Generating queries and answer descriptions

In this section we give an overview of the generation task for Quelo NLI. We start (Section 3.1) by describing the input to the generator. In Section 3.2, we motivate our choice for a grammar-based approach that enables the generation of fluent possibly multi-sentence text and paraphrases producing verbalisations from different perspectives. Next, we present the overall generation architecture (Section 3.3), describe in detail the surface realiser and its linguistic resources (Section 3.4) and illustrate how the generation approach supports fluency, versatility and portability (Section 3.5).

3.1 The input

The Query Tool formal framework (cf. [15]) defines a query as a labelled tree where edges are labelled with a relation name and nodes are labelled with a variable and a non-empty set of concept names from the ontology. Indeed, this is a representation of a *tree-shaped conjunctive query*. Figure 3 shows an example query tree.

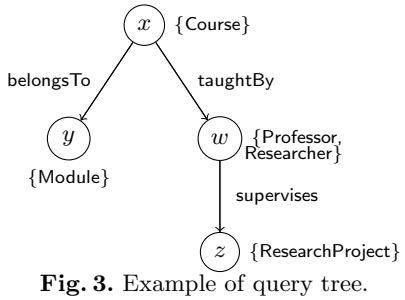

Fig. 3. Example of query tree.

Furthermore, each node of the query tree can be expressed as a concept of some Description Logic (DL) $\mathcal{L}$ using atomic concept instantiation, limited existential restriction and conjunction. Given a knowledge base $\mathcal{K}$ over a set of relations $\mathbf{R}$ and a set of concepts $\mathbf{C}$, then a concept in $\mathcal{L}$ is defined as $S ::= C \mid \exists R.(S) \mid S \sqcap S$ where $R \in \mathbf{R}$, $C \in \mathbf{C}$, $\sqcap$ denotes conjunction and $\exists$ is used for limited existential restrictions. For instance, the root node of the query tree in Figure 3 can be expressed as:

$$\mathsf{Course} \sqcap \exists\, \mathsf{belongsTo.Module} \sqcap \exists\, \mathsf{taughtBy.}\ (\mathsf{Professor} \sqcap \mathsf{Researcher} \sqcap \exists\, \mathsf{supervises.ResearchProject})$$

The input to the generator is thus a query tree[2] for query verbalisation and a query tree plus a set of selected nodes from the query for the generation answer descriptions.

3.2 Choice of a linguistically principled grammar framework

As mentioned in the introduction, most of the existing ontology and query verbalisers (e.g. the SWAT system [36] or Quelo Templates [8]) implement a deterministic mapping from ontology elements into natural language. These approaches rely either on templates or on a restricted grammar writing environment (DCGs). Following the consensus model assumption, they strictly associate concepts and relations with lexical categories (e.g. verb, noun, etc) and map each relation onto a clause permitting only a restricted syntactic variation (e.g. no relative clauses or Prepositional Phrases (PPs)). For instance, given the DL query (3a) and assuming a linearisation of that formula that matches the linear order it is presented in (see Section 3.3 for a description of query linearisation), a generator following these assumptions would produce a verbalisation such as (3b). By relaxing this strict deterministic mapping and allowing for additional syntactic constructions, more fluent verbalisations can be generated however. Thus the query verbalisation in (3c) aggregates the content into a single sentence using relative clauses, coordination and apposition.

(3) a. $\mathsf{WineGrape} \sqcap \exists \mathsf{hasVintageYear}.\mathsf{VintageYear} \sqcap \exists \mathsf{madeIntoWine}.[\mathsf{WhiteWine} \sqcap \mathsf{DryWine}]$
b. *I am looking for a wine grape. Its vintage year should be a vintage year. The wine grape should be made into a white wine. The white wine is a dry wine.*
c. *I am looking for a wine grape whose vintage year should be a vintage year and which should be made into a white wine, a dry wine.*

The assumed deterministic mapping also implies strict one-direction verbalisations. That is, the logical subject and object of a relation are mapped onto the syntactic subject and object of its verbalisation. For instance, given the DL query (4a) these verbalisers will produce a verbalisation such as the one in (4b) where the *car* is the subject of a clause. However, in some cases, the verbalisation of query answers changes the perspective of description, i.e. it shifts the topic of the query description to other entities in the set of answer tuples. For example, while the topic in the query (4b) is the *car* the emphasis in the answer descriptions in (4c) or (4d) is placed on the *car dealer*. The strict subject and object correspondence can also be relaxed by adding further syntactic constructions. The core meaning conveyed by the relation $\mathsf{soldBy}(\mathsf{Car}, \mathsf{CarDealer})$ can be rendered in NL in different ways. The verbalisation in (4c) uses a subject relative clause and active voice and (4d) a relative clause moving the by-agent to the subject position. They all describe the same situation but from different perspectives.

[2] For simplicity of description in this paper we assume a query tree as input for query verbalisation; however, as proposed in [21] query generation is incremental and the input is a given query tree plus a query revision operation.

(4) a. Car ⊓ ∃soldBy.CarDealer
b. *I'm looking for a car. It should be* ***sold by*** *a car dealer.*
c. *The car dealer which sells the car...*
d. *The car dealer by which the car is sold...*

To model a richer set of syntactic constructions, to support fluent verbalisations of KB queries and answer descriptions, we introduce a generation approach which relies on a non-deterministic grammar. This grammar permits the simple modelling of morpho-syntactic constraints (e.g. the long distance relative pronoun *which* in (3c) agrees with the inanimate noun WineGrape that it modifies) and the transparent implementation of the syntax semantics interface (e.g. the relation between CarDealer second argument of the binary relation and its surface position, for instance, as subject in (4c)). Concretely, we propose a generation approach based on a Feature Based Lexicalised Tree Adjoining Grammar (FB-LTAG, [32]) equipped with unification based compositional semantics ([12]). There are at least two major reasons for this choice. First, the grammar captures in a principled way the link between semantics, syntax and text and provides with standard algorithms. Second, it can be combined with an automatic lexicon extractor to provide a domain independent generation framework which produces verbalisations from any given KB regardless of its underlying vocabulary.

3.3 Overall natural language generation architecture

Starting from a query tree (Section 3.1) or query tree plus set of selected nodes, the generator follows a traditional NLG pipeline sequencing the following modules.

Document planning. In the query verbalisation process there is no content selection; the content to be verbalised is the whole query tree and is selected by the user through the query manipulation operations (Section 2). For the verbalisation of answer descriptions a subset of concept and relations from the query tree is selected (cf. [18] for a detail account on the selection mechanism).

As a first step an appropriate linear order is defined for the content expressed in a query tree. This linear order is given by a depth-first traversal of the tree plus a precedence order defined on the children of a node (as defined in [6,15]). Example (5a) shows a linearisation of the example query tree in Figure 3 following this method. Furthermore, to ensure the appropriate syntax/semantic interface, we make explicit the arguments of a relation using the variables associated with the nodes of the query tree (5b).

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

Surface realisation. To map the semantic content into a NL expression our surface realiser uses a FB-LTAG equipped with unification based compositional semantics and an automatic extracted lexicon for this grammar. Example (6) shows a surface realisation input-output pair, i.e. the input semantic (6a) and the output NL expression with underspecified NPs (6b).

(6) a. {*Course(x), taughtBy(e1,x,w), Professor(w)*}
b. *I am looking for* [$_{NP}$ *course*] *taught by* [$_{NP}$ *professor*].
c. *I am looking for a course taught by a professor.*

Referring expression generation. The referring expression module takes the output of the surface realiser as input, e.g. the NL expression in (6b). Based on a set of simple rules, it decides for each incomplete NP whether it should be verbalised as a pronoun, a definite or an indefinite noun phrase ([31,18]). We distinguish between first/subsequent mention of a discourse entity; and we use morpho-syntactic information associated to the syntactic context given by the relation in which the entity participates in as well as the noun verbalising the entity. The output of the referring expression module is a finalised NL expression, e.g. (6c) shows the output NL expression where both NPs were resolved using the indefinite determiner.

Here we concentrate on the surface realisation approach. In what follows, we describe in detail the surface realiser and the linguistic resources it uses, i.e. the grammar and the lexicon.

3.4 Surface realisation

We focus first on describing the grammar that our generator uses, i.e. FB-LTAG, and how the lexicon for this grammar is derived automatically from a given KB. We then summarise the algorithm used to generate with this grammar.

The grammar Informally, the FB-LTAG[3] consists in a set of pairs of *grammar units* of the form <tree schema, semantic schema>. The tree component is a phrase structure tree, namely elementary tree, whose nodes are decorated with feature structures. There are two types of elementary trees: initial and auxiliary. Initial trees are trees whose leaves are labeled with substitution nodes (marked with a down-arrow) or with terminal categories. Auxiliary trees are distinguished by a foot node (marked with a star) whose category must be the same as that of the root node. Two tree-composition operations are used to combine trees: substitution and adjunction. Substitution inserts a tree onto a substitution node of some other tree while adjunction inserts an auxiliary tree into a tree. The semantic component shares unification variables with the nodes of the tree thereby ensuring a correct mapping between syntactic and semantic arguments. As trees are combined, the semantics of the resulting derived tree is the union of their semantics modulo unification.

[3] For a more detailed introduction to TAG and FB-LTAG, see [32].

Figure 4 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 $\exists \mathsf{sell}.(\mathsf{Car} \sqcap \mathsf{Coupe})$.

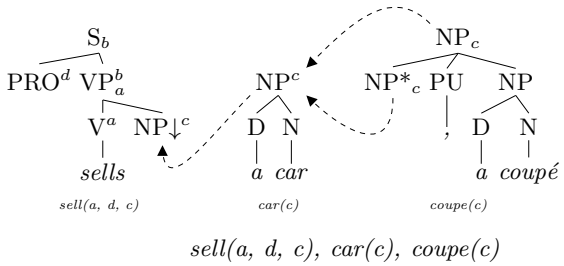

Fig. 4. Derivation and Semantics for "It sells a car, a coupé."

Each grammar unit describes a syntactic context (i.e. subcategorization frame plus surface realisation of the arguments) and relates it to a relation or concept of a KB. Figure 5 shows three grammar units describing different syntactic contexts. Both trees on the left correspond to the same subcategorization frame, namely a verb taking two Noun Phrases (NPs) as arguments, but they differ in the realisation of their arguments. The tree named nx0Vnx1 takes a canonical subject and object whereas the tree W0nx0Vnx1 takes a relative subject and a canonical object. The tree nx0Vpnx1 on the right belongs to a different subcategorisation frame, namely a verb that takes an NP and a PP argument.

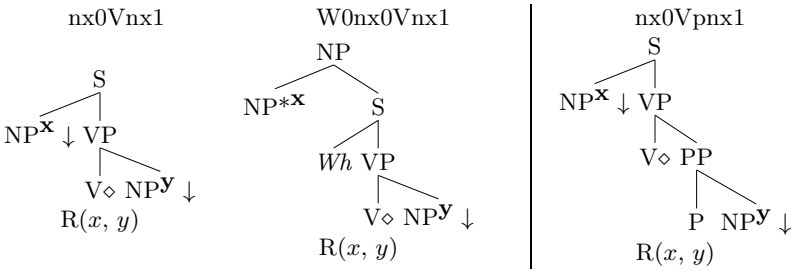

Fig. 5. <tree schema, semantic schema> pairs for nx0Vnx1 (e.g. *The car dealer sells a car*), W0nx0Vnx1 (e.g. *The car dealer which sells the car*) and nx0Vpnx1 (e.g. *The car runs on diesel*).

An FB-LTAG will usually associate several grammar units with a given lexical item. Grammar units sharing the same subcategorisation frame are grouped together into so-called *tree families*. Figure 6 (left) shows a small family of trees describing different syntactic contexts for the TrVerb_NPObj, i.e. verb taking two NP arguments, subcategorization class.

The FB-LTAG that Quelo NLI's generator uses consists of 110 grammar units describing the following syntactic contexts: active and passive, transitive verbs, adjectives, prepositional phrases, relative and elliptical clauses, gerund and participle modifiers.

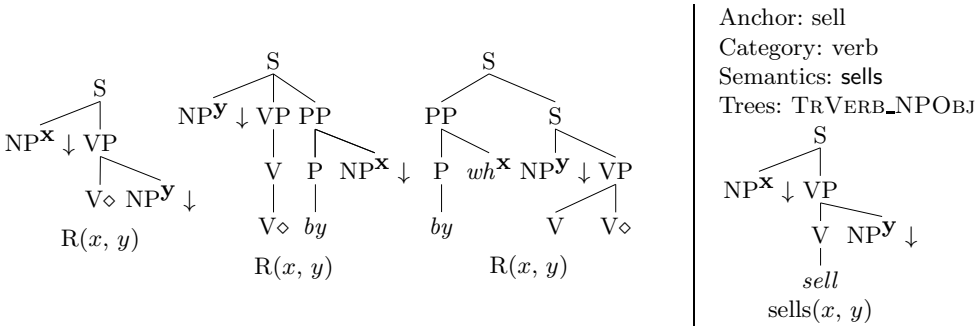

Fig. 6. LEFT. TrVerb_NPObj family: Active with canonical subject and object (a.), passive with canonical subject and canonical by-agent (b.) and passive with canonical subject and wh-by-agent (c.). RIGHT. Lexical Entry for the relation sells (top) and tree obtained when anchoring the tree schema (a.) with this lexical entry (bottom).

The lexicon The lexicon associates concepts and relations of a given KB with FB-LTAG grammar units. Each lexical entry specifies: the lexical item that will "anchor" the tree schema (Anchor), its lexical syntactic category (Category); the KB concept or relation (Semantics) and the tree (or set of trees) it is associated with (Trees). Figure 6 (right-top) shows a lexical entry for the KB relation sells. The Anchor and Semantics are used to instantiate tree schemas during generation. The Anchor specialises the anchor node (marked with $\diamond$) of the tree schema and the Semantics instantiates the predicate variable R. Figure 6 (right-bottom) shows the tree obtained by anchoring the tree schema (a.) in the left with the lexical entry for the relation sells.

The approach that we follow to automatically create such lexicon is derived from [31]. It builds on the fact that, in general, natural language is used to name KB concepts and relations ([25]). Thus, the idea is to analyse these names and try to predict complete syntactic contexts for them. Given a KB, the approach

proceeds in three steps to derive lexical entries for concepts and relations from their names as follows[4].

- Tokenization. First, relation(concept) names are tokenized in order to obtain a sequence of words. For instance, the relation name sold_by is tokenized as "sold by".
- POS tagging. Second, Trevisan's ([31]) customised Part-Of-Speech Tagger (POS Tagger) assigns tags from the Penn Treebank Project ([24]) to words in the tokenized sequences. The word sequence "sold by" is tagged as "sold:VBN by:IN".
- Mapping. Based on the sequence of tags assigned by the POS tagger and the interpretation of relations defined in ([31]), we associate relations(concepts) with one or more grammar units capturing a possible lexical and/or syntactic verbalisation thereof. This mapping is manually defined. Given the sequence "sold:VBN by:IN" the relation sold_by is associated with the grammar unit nx0VVVpnx1. In Table 1, we give some examples illustrating the mapping from POS sequences into grammar units of our grammar.

includes:VBZ	nx0VVnx1	NP_0 should include NP_1
run:VB on:IN	nx0VVpnx1	NP_0 should run on NP_1
equipped:VBN with:IN	nx0VVVpnx1	NP_0 should be equipped with NP_1
equipment:NN of:IN	nx0VVDNpnx1	NP_0 should be the equipment of NP_1
has:HVZ access:NN to:IN ac:NN	nx0VVNpnx1	NP_0 should have access to NP_1
has:HVZ address:NN	DNPnx0VVnx1	The address of NP_0 should be NP_1
is:BEZ suitable:JJ for:IN	nx0VVApnx1	NP_0 should be suitable for NP_1
photo:NN	DNPnx0VVnx1	the photo of NP_0 should be NP_1

Table 1. Example of Canonical Trees for each Subcategorisation Class

We have extracted lexicons for 9 ontologies from different domains: Quelo's built in Cars and Masters ontologies, Aquatic Resource Observation, GoodRelations, Wines, QALL-ME ontology (tourism domain), Adolena Ontology, Movies (TONES repository) and Camera OWL Ontology (Protégé repository). Each lexicon contains in average 680 lexical entries. There is in average 9 different syntactic contexts per lexical entry (i.e. lexical ambiguity). For instance, the KB relation sells is associated with 16 grammar units.

Surface realisation algorithm. For surface realisation, we adopt the chart-based surface realiser proposed in [14]. Given a FB-LTAG grammar and lexicon as described above and an input semantic formula, this realiser proceeds in four main steps:

[4] See [31] for a detailed account of the two first steps.

- Lexical Selection: retrieves from the grammar all grammar units whose semantics subsumes the input semantics.
- 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.
- Yield Extraction: all syntactically complete trees which are rooted in S/NP and associated with exactly the input semantics are retrieved. Their yields provide the set of generated (lemmatised) sentences/phrases.
- Morphological Realisation. Lexical lookup and unification of the features associated with lemmas in the generated lemmatised sentences yields the final set of output sentences.

Furthermore, the realisation algorithm is adapted (cf. [21]) for: Incremental generation (The verbalisation of each query refinement is based on the current query's associated chart state and a local change fired by a query refinement step –add, remove or delete some content to the current verbalisation.) and Order preserving generation (Verbalisations whose surface elements are as close in order as possible to the underlying query linearisation are favoured during generation).

3.5 How FB-LTAG supports fluency, versatility and portability

Fluent query verbalisations: Different verbalisations for different contexts of the same (type of) relation or concept. As mentioned in the previous section, the FB-LTAG lexicon associates KB concepts and relations with grammar units which represent different syntactic contexts. This not only relaxes the one-to-one data to NL mapping assumption by adding additional syntactic constructions, but also provides alternative syntactic contexts which might be required in different verbalisation contexts. This one-to-many mapping contributes to the generation of fluent verbalisations. As an example, the relation sold_by is associated with 16 different grammar units, Example (7) shows different verbalisations contexts of the relation which use different grammar units.

(7) a. *I am looking for a car sold by a car dealer.* (betanx0VPpnx1)
b. *I am looking for a car equipped with a car engine, located in a country and sold by a car dealer.* (betanx0ANDVPpnx1)
c. *I am looking for a car equipped with a manual gear transmission system. The car should run on a natural gas, should be sold by a car dealer and should be located in a country.* (betavx0ANDVVVpnx1)
d. *I am looking for a car whose make should be Toyota and which should be sold by a car dealer.* (ANDWHnx0VVVpnx1)
e. *I am looking for a car whose make should be Toyota and whose exterior color should be beige. The car should be sold by a car dealer.* (nx0VVVpnx1)
f. *A car dealer which sells the car (...).* (W1nx1Vnx0)
g. *A car dealer by which the car is sold (...).* (W1pnx1nx0VV)

Versatility: Different verbalisations from different perspectives. Amongst the different syntactic contexts associated by FB-LTAG lexicon with a relation(concept) name, some of them express the same situation but from a different perspective. This is the case of Examples (7f) and (7g). In Quelo NLI, this change of perspective occurs when the query answer descriptions are generated. Example (8a) shows a query in which instances of car make, equipment and car dealer are requested (cf. Section 2) and (8b) shows the description of the tuples obtained after executing this query. While in the query the topic is the *car*, the description of the results introduces the set of retrieved tuples that satisfies the query and thus, for instance, the topic for the 3rd component of the tuples is the *car dealer*.

(8) a. *I am looking for a car* ***produced by*** *a car make,* ***equipped with*** *an equipment and* ***sold by*** *a car dealer.*

b.

A car make which produces the car.	*A equipment with which that car is equipped.*	*A car dealer which sells that car.*
...	...	...

Portability: Verbalisations for different relations for ontologies of different domains. As stated in the introduction, a major requirement for Quelo NLI is portability. That is, it should be able to generate NL queries and query answer descriptions independently of the given KB. To support portability we combined a domain-independent hand-written grammar with a lexicon that is automatically derived from the KB (3.4: The lexicon).

We created manually a dataset of 164 queries from 9 ontologies from different domains, the proposed grammar-based approach was used to verbalise these queries. From this set, 145 queries where successfully verbalised, i.e. the coverage of the generation approach was 88.4%. With the verbalised queries, we created a corpus, namely QQueryCorpus, of <DL query, NL query> pairs[5]. This corpus covers 158 different relations names from the 9 ontologies. Example (9) shows the verbalisation for 2 of these relations which belong to different relation classes. Example (10), shows verbalisations of 3 relations from different ontologies that belong to the same relation class.

(9) a. *I am looking for a dessert course whose drink should be a sweet riesling.*
has**H**VZ: Drink:**N**N (Wines)

b. *I am looking for an event which should be in a site.*
is:**B**EZ In:**I**N Site:**N**N (QALL-ME)

(10) a. *I am looking for an aquatic resource found in a fao fishing area.*
found:**V**BN in:**I**N (Aquatic Resource Observation)

b. *I am looking for a wine made from a wine grape.*
made:**V**BN from:**I**N (Wines)

c. *I am looking for a student enrolled at a university.*
enrolled:**V**BN at:**I**N (masters)

[5] Go to `http://www.loria.fr/perezlla/content/software/` for access to this corpus.

4 Discussion

The generation approach that we proposed satisfies two important requirements of Quelo NLI. First, the grammar models in a principled way different syntactic contexts necessary for the fluent verbalisation of queries and answer descriptions. Second, its combination with an automatically extracted lexicon provides a domain independent framework.

We carried out a human-based evaluation experiment ([21]) aimed at comparing the template and grammar-based generation approaches. This evaluation showed that the fluency given by additional syntactic constructions in the grammar-based approach results in query verbalisations that are better accepted by users. The next step will consist in the design of a task-based evaluation experiment. This will aim at assessing the grammar-based generation approach as well as comparing Quelo NLI approach with other query interfaces (e.g. free text query writing) based on the outcome of the task.

Building on this generation framework, future work could explore strategies for further improving the current approach to lexicalisation of ontology relations(concepts) following ideas from ([33,28]). Alternative lexicalisations to those that are built from the words used in relation(concept) names could be learnt (semi-) automatically. For instance, the lexicalisation *with* for the relation ofStdAbundanceLevel, (11a) shows a query and (11b) a sentence from the ontology documentation[6]. The link between text and the underlying query semantics that Quelo NLI implements supports these kind of alternative lexicalisations without suffering of vocabulary discrepancy problems.

(11) a. AquaticResourceObservation $\sqcap \exists$ isObservationOf.(FAOFishingArea $\sqcap \exists$ ofStdAbundanceLevel.StdAbundanceLevel)

b. *The aquatic resource for Skipjack tuna in Northern Atlantic in 2004 (as reported in 2008) was observed with low abundance level.*

References

1. Androutsopoulos, I., Ritchie, G. D., and Thanisch, P. Natural language interfaces to databases–an introduction. *Natural language engineering 1*, 01 (1995), 29–81.
2. Benamara, F., and Saint Dizier, P. Webcoop: A cooperative question-answering system on the web. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2* (2003), Association for Computational Linguistics, pp. 63–66.
3. Bontcheva, K. Generating tailored textual summaries from ontologies. In *The Semantic Web: Research and Applications*. Springer, 2005, pp. 531–545.
4. Bouayad-Agha, N., Casamayor, G., Wanner, L., Díez, F., and Hernández, S. L. Footbowl: Using a generic ontology of football competition for planning match summaries. In *The Semantic Web: Research and Applications*. Springer, 2011, pp. 230–244.

[6] Taken from http://ontologydesignpatterns.org/wiki/Submissions:AquaticResourceObservation

5. Damljanovic, D., Agatonovic, M., and Cunningham, H. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In *Proceedings of the 7th International Conference on The Semantic Web: Research and Applications - Volume Part I* (Berlin, Heidelberg, 2010), ESWC'10, Springer-Verlag, pp. 106–120.
6. Dongilli, P. Natural language rendering of a conjunctive query. *KRDB Research Centre Technical Report No. KRDB08-3). Bozen, IT: Free University of Bozen-Bolzano 2* (2008), 5.
7. Franconi, E., Guagliardo, P., and Trevisan, M. An intelligent query interface based on ontology navigation. In *Proceedings of the Workshop on Visual Interfaces to the Social and Semantic Web (VISSW 2010)* (2010), vol. 565, Citeseer.
8. Franconi, E., Guagliardo, P., and Trevisan, M. Quelo: a nl-based intelligent query interface. In *Pre-Proceedings of the Second Workshop on Controlled Natural Languages* (2010), vol. 622.
9. Franconi, E., Guagliardo, P., Trevisan, M., and Tessaris, S. Quelo: an Ontology-Driven Query Interface. In *Description Logics* (2011).
10. Furnas, G. W., Landauer, T. K., Gomez, L. M., and Dumais, S. T. The vocabulary problem in human-system communication. *Commun. ACM 30*, 11 (Nov. 1987), 964–971.
11. Galanis, D., and Androutsopoulos, I. Generating multilingual descriptions from linguistically annotated owl ontologies: the naturalowl system. In *Proceedings of the Eleventh European Workshop on Natural Language Generation* (2007), Association for Computational Linguistics, pp. 143–146.
12. Gardent, C. Integrating a unification-based semantics in a large scale Lexicalised Tree Adjoining Grammar for French. In *COLING'08* (Manchester, UK, 2008).
13. Gardent, C., Banik, E., and Perez-Beltrachini, L. Natural Language Generation and Interfaces to Knowledge Bases. Invited Tutorial at Knowledge Capture (K-CAP), Banff (Canada), 2011.
14. Gardent, C., and Perez-Beltrachini, L. RTG based surface realisation for TAG. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)* (Beijing, China, August 2010), Coling 2010 Organizing Committee, pp. 367–375.
15. Guagliardo, P. Theoretical foundations of an ontology-based visual tool for query formulation support. Master's thesis, KRDB Research Centre, Free University of Bozen-Bolzano, October 2009.
16. Hallett, C., Scott, D., and Power, R. Composing questions through conceptual authoring. *Computational Linguistics 33*, 1 (2007), 105–133.
17. Hixon, B., and Passonneau, R. J. Open dialogue management for relational databases. In *HLT-NAACL* (2013), pp. 1082–1091.
18. Juárez-Castro, X. I. Presenting query results in natural language for quelo. Tech. rep., Free University of Bozen-Bolzano, 2014.
19. Kaljurand, K., and Fuchs, N. E. Verbalizing owl in attempto controlled english. In *OWLED* (2007), vol. 258.
20. Kaufmann, E., and Bernstein, A. *How useful are natural language interfaces to the semantic web for casual end-users?* Springer, 2007.
21. Laura Perez-Beltrachini, C. G., and Franconi, E. Incremental query generation. In *EACL 2014* (Gothenburg, Sweden, April 2014).
22. Liang, S. F., Scott, D., Stevens, R., and Rector, A. Unlocking medical ontologies for non-ontology experts. In *Proceedings of BioNLP 2011 Workshop* (2011), Association for Computational Linguistics, pp. 174–181.

23. Lopez, V., Fernández, M., Motta, E., and Stieler, N. Poweraqua: Supporting users in querying and exploring the semantic web. *Semantic Web 3*, 3 (2012), 249–265.
24. Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist. 19*, 2 (June 1993), 313–330.
25. Mellish, C., and Sun, X. The semantic web as a linguistic resource: Opportunities for natural language generation. *Knowledge-Based Systems 19*, 5 (2006), 298–303.
26. Power, R. Towards a generation-based semantic web authoring tool. In *Proceedings of the 12th European Workshop on Natural Language Generation* (2009), Association for Computational Linguistics, pp. 9–15.
27. Power, R. Complexity assumptions in ontology verbalisation. In *Proceedings of the ACL 2010 Conference Short Papers* (2010), Association for Computational Linguistics, pp. 132–136.
28. Snchez, D., Moreno, A., and Vasto-Terrientes, L. D. Learning relation axioms from text: An automatic web-based approach. *Expert Systems with Applications 39*, 5 (2012), 5792 – 5805.
29. Tablan, V., Damljanovic, D., and Bontcheva, K. A natural language query interface to structured information. In *Proceedings of the 5th European Semantic Web Conference on The Semantic Web: Research and Applications* (Berlin, Heidelberg, 2008), ESWC'08, Springer-Verlag, pp. 361–375.
30. Tennant, H. R., Ross, K. M., Saenz, R. M., Thompson, C. W., and Miller, J. R. Menu-based natural language understanding. In *Proceedings of the 21st annual meeting on Association for Computational Linguistics* (1983), Association for Computational Linguistics, pp. 151–158.
31. Trevisan, M. A portable menuguided natural language interface to knowledge bases for querytool. Master's thesis, Free University of Bozen-Bolzano (Italy) and University of Groningen (Netherlands), 2010.
32. Vijay-Shanker, K., and Joshi, A. K. Feature structures based tree adjoining grammars. In *Proceedings of the 12th conference on Computational linguistics-Volume 2* (1988), Association for Computational Linguistics, pp. 714–719.
33. Walter, S., Unger, C., and Cimiano, P. A corpus-based approach for the induction of ontology lexica. In *Natural Language Processing and Information Systems*. Springer, 2013, pp. 102–113.
34. Waltz, D. L. An english language question answering system for a large relational database. *Communications of the ACM 21*, 7 (1978), 526–539.
35. Webber, B., Joshi, A., Mays, E., and McKeown, K. Extended natural language data base interactions. *Computers & Mathematics with Applications 9*, 1 (1983), 233 – 244.
36. Williams, S., and Power, R. Grouping axioms for more coherent ontology descriptions. In *Proceedings of the 6th International Natural Language Generation Conference* (2010), Association for Computational Linguistics, pp. 197–201.