Incremental Query Generation

Claire Gardent
(Joint work with Laura Perez-Beltrachini)

L. Perez-Beltrachini and C. Gardent
Incremental Query Generation
EACL 2014. Gothenburg, Sweden, April 2014.

NL-Based Query Interface

KBs are widely used

- to integrate heterogeneous data sources
- to support user queries

... but the formal languages used to implement them (OWL, RDF) are difficult to handle for non expert users

NL query interfaces provide a means to develop user frienly query interfaces to KBs

Existing Approaches (e.g., Conceptual Authoring, Quelo, SWAT)

Templates or Restricted Grammar Writing Environnment (DCGs)

Ad hoc Generation Algorithms

“Consensus Model”: one clause per relation; strict matching between concept/relations and linguistic categories; restricted syntactic variation (e.g., no relative clauses or PPs)

Quelo, a NL-Based Query Interface

Quelo is a NL interfaces for KBs which was developed by the University of Bolzano (E. Franconi)

Requires neither KL of the formal KR languages nor of the KB content

At each point during querying, Quelo

- computes all extensions of the current query that are consistent and non redundant (using automated reasoners)
- displays a NL description of these extensions
- allows for revisions of the current query

Example Queries and Verbalisations

(1) a. $\top$ (initial query)
I am looking for **something**

b. *Man* (substitute concept)
I am looking for **a man**

c. $\mathit{Man} \sqcap \mathit{Young}$ (add compatible concept)
I am looking for a young **man**

d. $\mathit{Man} \sqcap \mathit{Young} \sqcap \exists \mathit{isMarried}.(\mathit{Person})$ (add relation)
I am looking for a young **man who is married to a person**

e. $\mathit{MarriedMan} \sqcap \mathit{Young}$ (substitute selection)
I am looking for a young **married** man

f. *MarriedMan* (delete concept)
I am looking for a married man

Quelo 2

Templates or Restricted Grammar Writing Environnment (DCGs)

Ad hoc Generation Algorithms

“Consensus Model”: one clause per relation; strict matching between concept/relations and linguistic categories; restricted syntactic variation (e.g., no relative clauses or PPs)

Quelo 2

Templates or Restricted Grammar Writing Environnment (DCGs)
$\Rightarrow$ FB-LTAG

Ad hoc Generation Algorithms

“Consensus Model”: one clause per relation; strict matching between concept/relations and linguistic categories; restricted syntactic variation (e.g., no relative clauses or PPs)

Quelo 2

Templates or Restricted Grammar Writing Environnment (DCGs)
$\Rightarrow$ FB-LTAG

Ad hoc Generation Algorithms
$\Rightarrow$ Tabular Algorithm, Beam Search

“Consensus Model”: one clause per relation; strict matching between concept/relations and linguistic categories; restricted syntactic variation (e.g., no relative clauses or PPs)

Quelo 2

Templates or Restricted Grammar Writing Environnment (DCGs)
$\Rightarrow$ FB-LTAG

Ad hoc Generation Algorithms
$\Rightarrow$ Tabular Algorithm, Beam Search

“Consensus Model”: one clause per relation; strict matching between concept/relations and linguistic categories; restricted syntactic variation (e.g., no relative clauses or PPs)
$\Rightarrow$ Standard Syntax (relative clauses, PPs, ellipses, Left Node Raising etc)

NL Generation in Quelo

Inputs: the current query Q + the user's update of that query U
Tree shaped conjunctive formulae in Description Logic (DL)

Output: NL verbalisation of Q updated with U

Constraints on the Generation Algorithm

- should support incrementality (revisions, deletions and additions)
- should preserve order

Preserving Order

$Car \sqcap \exists runOn.(Diesel) \sqcap \exists equippedWith.(AirCond)$

✓ A car which runs on Diesel and is equipped with air conditioning

$\times$ A car which is equipped with air conditioning and runs on Diesel

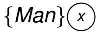
The NLG Architecture

Document planning: linearises the input query and partitions the input into sentence size chunks

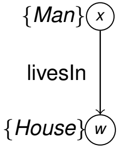
Surface realisation: maps each sentence size $\mathcal{L}$ formula into a sentence

Referring expression generator: verbalises NPs.

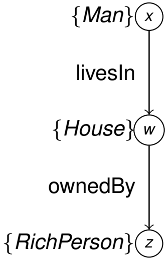
Example Query Sequence



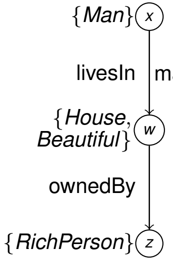
(a)



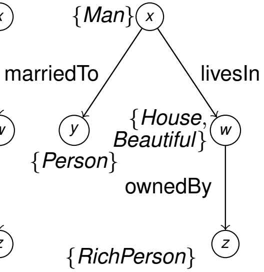
(b)



(c)

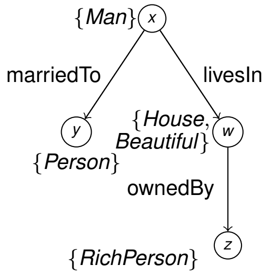


(d)



(e)

Query Linearisation (depth-first) and Translation



Man marriedTo Person livesIn House Beautiful ownedBy RichPerson

{ *Man(m), marriedTo(m,p), Person(p), livesIn(p,h), House(h), Beautiful(h), ownedBy(h,r), RichPerson(r)* }

Document Planning (Query Segmentation)

Given a linearised query q, the document planner uses some heuristics based on the number and the types of relations/concepts present in q to output a sequence of sub-formulae each of which will be verbalised as a sentence.

Surface Realisation

Grammar-Based (hand-written)

- No training corpus
- Restricted variations (conjunction of binary relations and properties)
- Portable to any KB

Lexicon automatically extracted from KB (M. Trevisan 2010)

Surface Realisation (Ct'd)

Tabular Algorithm

- Efficient (avoids recomputation of intermediate structures)
- Simple implementation of revisions (addition, deletion, substitution) operations

Beam Search

- Efficient (avoids recomputation of intermediate structures)
- Cost function to preserve linear order

Incremental Generation

C, the current chart. *A*, an empty agenda.

Add concept or property X: the trees selected by *X* are added to *A* and tried for combination with the elements of *C*.

Substitute selection X with Y : all chart items derived from a tree selected by *X* are removed from the chart. Conversely, all chart items derived from a tree selected by *Y* are added to the agenda and tried for combination with the elements of *C*.

Delete selection X: all chart items derived from a tree selected by *X* are removed from *C*. Intermediate structures that had previously combined with a tree selected by *X* are moved to the agenda and the agenda is processed until generation halts.

Beam Search

Scoring function favors derivations with low word order cost and large semantic coverage

Word Order Cost = distance between actual position and required position (given by the linearised input)

Semantic Coverage = number of literals covered by derivation

Referring Expression Generation

Input: Sequence of phrase structure trees output by the surface realiser

Uses heuristics to decide for each NP whether it should be verbalised as a pronoun, a definite or an indefinite NP.

Heuristics based on the linear order and the morpho-syntactic information contained in the phrase structure trees of the generated sentences.

Evaluation

Linearisation: Does the scoring mechanism ensure that the generated queries respect the strict total order of the query tree linearisation ?

Better than templates? Does our grammar based approach produce more fluent and less ambiguous NL query than the initial template based approach currently used by Quelo ?

Portability. Does the automatic extraction of lexicons from ontology support generic coverage of arbitrary ontologies ?

Linearisation

Does the word order in the generated queries match the linearisation of the input query tree ?

4 series of queries $q_1 \dots q_n$ where q_{i+1} is an increment of q_i. 14 revisions in total

encompass addition, deletion and substitution of possible operations at different points of the preceding query (the last node/edge or node/edge occurring further to the left of the previous query)

For all queries, the word order produced by the generator matches the linearisation of the DL query.

Template vs. Grammar-Generated Queries

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.

I am looking for a car whose make is a Land Rover, whose body style is an off-road car and whose exterior color is beige.

Assessing Quelo Template-Based Queries

41 queries capturing different combinations of concepts and relations

8 raters

50% of the queries are rated as disfluent
10% of the queries are rated as unclear

Free Comments: too repetitive, lacks aggregation

Comparing Template-Based and Grammar-Generated Queries

10 raters, 14 query pairs built from two ontologies (cars, universities)

	Fluency	Clarity
Grammar	**19.76**	6.87
Templates	7.2	8.57

Portability

General, domain independent, grammar + Automatically extracted lexicon (cf. Trevision 2010)

Lexicon extraction tested on 200 ontologies. Coverage: 85% of the ontology relations (12000 relns, 13 templates)

40 queries on 5 ontologies (cinema, wines, human abilities, assistive devices, ecommerce). Coverage 87%

Conclusion

Previous approaches: ad hoc generation algorithms based on templates or restricted grammars (DCG)

Tabular algorithm naturally supports the definition of an incremental algorithm for query verbalisation

The grammar based approach generates queries that are better accepted by human users

Future Work

Improve fluency, clarity (lexicon extraction, SR ranking)

Interaction between input segmentation, surface realisation and referring expression generation

Use existing system to build a parallel corpus (DL/NL query) and train

- joint model of input segmentation, surface realisation and referring expression generation
- ranking module (to guide beam search)

Thanks ! Questions ?