An extension of SPARQL for expressing qualitative preferences

A. Troumpoukis[1,2] S. Konstantopoulos[1] A. Charalambidis[1]

[1] Institute of Informatics and Telecommunications,
NCSR 'Demokritos', Athens, Greece
{antru,konstant,acharal}@iit.demokritos.gr

[2] Department of Informatics and Telecommunications,
University of Athens, Greece

25 October 2017, Vienna, Austria

Preferences

Preferences are being studied in Artificial Intelligence, Machine Learning and Databases.

Preferences on the Semantic Web

- RDF data: varying trustworthiness, quality and relevance
- New query languages: improve usability of such data

Integration of Preferences in Queries

- more faithful to what a user intends to say
- too many answers $\leadsto$ more-preferred results
- no answers $\leadsto$ less-preferred results

Quantitative vs. Qualitative preferences

Main approaches for representing preferences:

- *The quantitative approach:* as degrees of interest ("My preference in science-fiction books is 0.8 while in novels 0.4").
- *The qualitative approach:* by direct comparisons ("I like action movies more that comedies").

Qualitative approach

- more general (preference function cannot always be defined)
- more intuitive ("humans are rarely willing to express their preferences directly in terms of a value function.")

Chomicki Framework

The qualitative approach in databases

An influential work in this area is: J. Chomicki, "*Preference Formulas in Relational Queries*", ACM TODS, 2003.

Main ideas:

- Preferences between tuples are specified using *binary preference relations*, defined using logic formulas.
- A new relational algebra operator is introduced, that eliminates from its argument relation the less preferred tuples according to the given preference relation.

Chomicki Framework (cont.)

Example

"I prefer one `movie` over another iff their genre is the same and the first one has a longer duration".

$$(i, t, g, d) \succ_P (i', t', g', d') \equiv (g = g') \wedge (d > d').$$

Movies table

ID	Title	Genre	Duration
m_1	A New Hope	Sci-fi	121
m_2	The Empire Strikes Back	Sci-fi	124
m_3	Return of the Jedi	Sci-fi	130
m_4	Die Hard	Action	131
m_5	Die Hard with a Vengeance	Action	128

Chomicki Framework (cont.)

Example

"I prefer one `movie` over another iff their genre is the same and the first one has a longer duration".

$$(i, t, g, d) \succ_P (i', t', g', d') \equiv (g = g') \wedge (d > d').$$

Movies table

Winnow result

ID	Title	Genre	Duration
m_1	A New Hope	Sci-fi	121
m_2	The Empire Strikes Back	Sci-fi	124
m_3	Return of the Jedi	Sci-fi	130
m_4	Die Hard	Action	131
m_5	Die Hard with a Vengeance	Action	128

Our approach

SPREFQL

- Extends SPARQL with a preference solution modifier

Preference Solution Modifier

- Operates after Projection.
- Expresses a *binary preference relation* between query solutions.
- Eliminates the solutions that are less preferred according to the given preference relation.

A simple SPREFQL query

Example

"I prefer one `movie` over another iff their genre is the same and the first one has a longer duration".

SPARQL Query

```
SELECT ?title ?genre ?runtime WHERE {
  ?s a :film.  ?s :title ?title.
  ?s :genre ?genre.  ?s :runtime ?runtime.
}
```

A simple SPREFQL query

Example

"I prefer one `movie` over another iff their genre is the same and the first one has a longer duration".

SPREFQL Query

```
SELECT ?title ?genre ?runtime WHERE {
  ?s a :film.  ?s :title ?title.
  ?s :genre ?genre.  ?s :runtime ?runtime.
}
PREFER (?title1 ?genre1 ?runtime1)
TO     (?title2 ?genre2 ?runtime2)
IF (?genre1 = ?genre2 && ?runtime1 > ?runtime2)
```

A simple SPREFQL query

Example

"I prefer one `movie` over another iff their genre is the same and the first one has a longer duration".

SPREFQL Query

```
SELECT ?title ?genre ?runtime WHERE {
  ?s a :film.  ?s :title ?title.
  ?s :genre ?genre.  ?s :runtime ?runtime.
}
PREFER (?title1 ?genre1 ?runtime1)
TO     (?title2 ?genre2 ?runtime2)
IF (?genre1 = ?genre2 && ?runtime1 > ?runtime2)
```

A simple SPREFQL query

Example

"I prefer one `movie` over another iff their genre is the same and the first one has a longer duration".

SPREFQL Query

```
SELECT ?title ?genre ?runtime WHERE {
  ?s a :film.  ?s :title ?title.
  ?s :genre ?genre.  ?s :runtime ?runtime.
}
PREFER (?title1 ?genre1 ?runtime1)
TO     (?title2 ?genre2 ?runtime2)
IF (?genre1 = ?genre2 && ?runtime1 > ?runtime2)
```

Another SPREFQL query

- Any FILTER constraint can also appear as IF constraint
- e.g. EXISTS expression
- IF constraints might also be *extrinsic*, referring to data not already fetched to satisfy the WHERE clause

Example

"I prefer original movies to their sequels".

SPREFQL Query

```
SELECT ?film ?title WHERE { ... }
PREFER (?film1 ?title1)
TO     (?film2 ?title2)
IF EXISTS { ?film1 :sequel ?film2 }
```

Preference Compositions in SPREFQL

- `AND` combinator: the two preferences are of equal importance

SPREFQL Query

```
SELECT ... WHERE {...}
PREFER (...) TO (...)
IF (...) AND (...)
```

- `PRIOR TO` combinator: the first preference is more important

SPREFQL Query

```
SELECT ... WHERE {...}
PREFER (...) TO (...)
IF (...) PRIOR TO (...)
```

Expressive Power of SPREFQL

- SPREFQL queries can be rewritten into SPARQL 1.1

SPREFQL Query

```
SELECT L WHERE {
  P
}
PREFER L1 TO L2
IF C
```

SPARQL Query

```
SELECT L WHERE {
  P
  FILTER NOT EXISTS {
    P'
    FILTER C'
  }
}
```

- Separate definitions of preferences from hard constraints
- Smaller queries, more human readable
- SPREFQL syntax allows easily recognize opportunities to apply specialized algorithms (such as BNL)

The BNL algorithm

- BNL identifies preferred solutions on a single run, avoiding the quadratic comparison
- But: The preference relation must be transitive, otherwise the algorithm might admit solutions that are not the most preferred ones

Evaluation

	query execution time (in ms)		
	QB	**RW**	**BNL**
Q1	556	4,750	812
Q2	52	188	65
Q3	52	254	91
Q4	872	197,044	1,238
Q5	872	193,338	2,370
Q6	135	296	–
Q7	85	93	–

QB Query Base (w/o preference)

RW Equivalent SPARQL 1.1 Query, NOT EXISTS transformation.

BNL SPREFQL query execution using BNL algorithm. Not applicable in Q6-Q7.

Ongoing and future work

- Implementation in rdf4j
- Deployable using docker-compose
- Live Demo

Most important future work:

- Get some of you to help me with user study
- Experienced with SPARQL

Thank you for your attention!

More about SPREFQL at

http://bitbucket.org/dataengineering/sprefql