Search by Strategy

Roberto Cornacchia - Spinque

Search Solutions – London - October 21, 2010
A simple search task

A real-estate search scenario

- Location: Amsterdam
- Price range: 220.000 - 250.000 €
- Size: 75+m²

Many real-estate search engines can satisfy this easy query

So under-specified, that it is likely to be nearly useless (www.funda.nl gives back ~350 results)

Most available engines do allow some more complexity

How much more?
A more desirable query

• Location: Amsterdam
  Near the historical canals, but far from high-crime districts

• Price range: 220.000 - 250.000 €
  Hey, 252.000 € is still fine, of course!

• Size: I'm not sure, let me choose later

• With backyard
  or with balcony that faces west

When is the benefit of building (and updating!) complex domain-specific search engines worth the effort?
Spinque

- Don't build the ultimate search engine for all possible queries

- “Draw” a number of search strategies

- Click. **Generate** Web search engines on probabilistic DB
Full-text search: a 3-block strategy

All houses

Amsterdam, canals

Rank full-text
The simplest search engine

Spinque!
Blocks have parameters

- All houses
- Keywords
- Rank full-text

= has free parameters
Parameters become UI controls

Keywords
amsterdam canals balcony faces west

Spinque!
Result streams can be combined...

Location → Rank on location → Union

All houses → Union

Keywords → Rank full-text → Union

= has free parameters
... and weighted

Keywords

amsterdam canals balcony faces west
Filtering result streams

- Location
  - Rank on location
- All houses
  - Select on attribute
- Keywords
  - Rank full-text
- Union

= has free parameters
I'd love a backyard, but not a must

Keywords:
amsterdam canals balcony faces west|

Desired attributes:

Location:

Spinque!
Excluding results

- Location
  - Rank on location
- Crime map
  - Rank on location
- All houses
  - Select on attribute
- Keywords
  - Rank full-text

Difference

Union

= has free parameters
Maybe not in the red light district..

Keywords
amsterdam canals balcony faces west

Desired attributes

Location

Exclude high-crime locations

Spinque!
Not so sure about the rest

Keywords

amsterdam canals balcony faces west|

Desired attributes

Location

Facets can be used to filter, but also to re-rank
Intellectual property search

- Large corpora, structured / unstructured data, multi-lingual
- IP search professionals perform complex searches
- Flexibility in drawing and updating strategies
- Transparency of results
- Probabilistic reasoning avoids manual merging of several boolean searches

"Prior Art Candidates" Strategy

Select reference patent

Patent corpus

Same assignee

Same IPCR classifications

Mix and rank with keywords from reference patent

Patents in the same family
How can strategies help?

- Strategies encapsulate domain expert knowledge \((\text{how to find})\)
- Strategies abstract away search expert knowledge \((\text{how to search})\)
- Results streams can be analysed / debugged at any step
- Strategies can be stored / shared / improved / published
- Strategies can mix exact (DB) and ranked (IR) searches
- No “human (probabilistic) join” needed
What's inside building blocks?

A = Select [attr='backyard'] (All_houses)
B = Join INDEPENDENT [id=id] (Input_houses,A)
RESULT = Project DISTINCT [id] (B)

- A BB encapsulates the **search expert knowledge**
- In principle, any language (scripting languages, SQL, XQuery, etc)
- Spinque uses a subset of Probabilistic Relational Algebra (PRA) [Fuhr, Rölleke 97]
What happens to building blocks?

-- A = Select [attr='backyard'] (All_houses)
Create view A as
Select *
From All_houses
Where attr='backyard';

-- B = Join INDEPENDENT [id=id] (Input_houses,A)
Create view B as
Select I.id, A.id, I.prob * A.prob as prob
From Input_houses as I, A
Where I.id = A.id;

-- RESULT = Project DISTINCT [id] (B)
Create view RESULT as
Select id, 1-prod(1-prob) as prob
From B
Group by id;

• This encapsulates the DB expert knowledge
• Spinque translates PRA to standard SQL and fires it to a DB engine
Isn't SQL too slow for IR?!

- Traditional relational DB engines carry outdated design principles

**Storage model**
- Traditional: row-oriented, best for OLTP
- Next-generation: column-oriented, best for OLAP

**CPU & RAM**
- Traditional: assume CPU & RAM are fast, no optimisation
- Next-generation: cache-conscious, data-hungry algorithms

- MonetDB, Ingres/Vectorwise
  - *Flexible and efficient IR using Array Databases*, Cornacchia et al., VLDBJ 2008
Summary

- Spinque promotes a **separation of concerns** in the engineering of search engines
  - Search strategy UI: *automatised*
  - Search strategy definition: the real *focus*, make it easy and flexible
  - Search strategy processing: *automatised*

- Spinque promotes **IR + DB**
  - IR + DB computed on a relational database engine – no human joins
  - IR + DB optimised within the very same framework – no black boxes
  - Modern DB technology *can* support IR efficiently
Thank you