

Semantic Search

Algorithmic Problems Around the Web #8

Yury Lifshits

<http://yury.name>

CalTech, Fall'07, CS101.2, <http://yury.name/algoweb.html>

The challenge of the Semantic Web, therefore, is to provide a language that expresses both data and rules for reasoning about the data and that allows rules from any existing knowledge representation system to be exported onto the Web.

*T. Berners-Lee, J. Hendler, O. Lassila
Semantic Web, 2001*

Outline

- 1 Introduction to Semantic Web
 - Concept and History of Development
 - Architecture of Semantic Web
 - Concept of Semantic Search

Outline

- 1 Introduction to Semantic Web
 - Concept and History of Development
 - Architecture of Semantic Web
 - Concept of Semantic Search
- 2 Three Algorithms for Semantic Search
 - Minimal Answers
 - Concept Matching
 - Computing Interconnections

Outline

- 1 Introduction to Semantic Web
 - Concept and History of Development
 - Architecture of Semantic Web
 - Concept of Semantic Search
- 2 Three Algorithms for Semantic Search
 - Minimal Answers
 - Concept Matching
 - Computing Interconnections
- 3 Directions for Further Research

Part I

Sematic Web

What is it?

What is already done?

What remains to be done?

Motivating Scenarios

A person asking his web-agent:

- Book the ticket for the movie “The Lives of Others” in the nearest cinema that shows it today evening

Motivating Scenarios

A person asking his web-agent:

- Book the ticket for the movie “The Lives of Others” in the nearest cinema that shows it today evening
- Find a suitable wine for every item in this menu. If possible, choose French

Motivating Scenarios

A person asking his web-agent:

- Book the ticket for the movie “The Lives of Others” in the nearest cinema that shows it today evening
- Find a suitable wine for every item in this menu. If possible, choose French
- Microwave, please, go to the website of the dish manufacturer and download the optimal parameters for cooking

Timeline

- **1994:** Foundation of W3C. They develop standards such as: HTML, URL, XML, HTTP, PNG, SVG, CSS
- **1998:** Tim Berners-Lee published “Semantic Web Road Map”
- **1999:** W3C launched groups for designing Semantic Web foundations, the first version of RDF is published
- **2000:** American defence research institution started investigations for ontology descriptions (DAML+OIL project)
- **2001:** “The Semantic Web” paper in Scientific American
- **2004:** New version of RDF, ontology description language OWL
- **2006:** Candidate recommendation of SPARQL, a query language for Semantic Web

Naïve Plan

- 1 Develop a MEGA-language that is powerful enough to describe all human knowledge and is machine understandable at the same time.
- 2 Force all web publishers translate their websites to this language
- 3 Write programs that can search in and reason about all the information in the web

Naïve Plan

- 1 Develop a MEGA-language that is powerful enough to describe all human knowledge and is machine understandable at the same time.
- 2 Force all web publishers translate their websites to this language
- 3 Write programs that can search in and reason about all the information in the web

There is a more practical solution for the first step

RDF and OWL

Tim Berners-Lee suggested to **separate** development of syntax and semantic of this MEGA-language:

Resource Description Framework (**RDF**) is a syntax for documents of Semantic Web. It uses links to **ontologies**

Ontology Web Language (**OWL**) is a language for ontology description

RDF and OWL

Tim Berners-Lee suggested to **separate** development of syntax and semantic of this MEGA-language:

Resource Description Framework (**RDF**) is a syntax for documents of Semantic Web. It uses links to **ontologies**

Ontology Web Language (**OWL**) is a language for ontology description

Ontology describes classes of objects, their properties and relationships in some domain, e.g. toy shops

Semantic Web Step-by-Step

- 1 Syntax for knowledge representation (done: RDF)

Semantic Web Step-by-Step

- 1 Syntax for knowledge representation (done: RDF)
- 2 Ontology description language (done: OWL)

Semantic Web Step-by-Step

- 1 Syntax for knowledge representation (done: RDF)
- 2 Ontology description language (done: OWL)
- 3 Web-services description language (started: OWL-S)

Semantic Web Step-by-Step

- 1 Syntax for knowledge representation (done: RDF)
- 2 Ontology description language (done: OWL)
- 3 Web-services description language (started: OWL-S)
- 4 Tools for reading/publishing Semantic Web documents (started: Jena, Haystack, Protege)

Semantic Web Step-by-Step

- 1 Syntax for knowledge representation (done: RDF)
- 2 Ontology description language (done: OWL)
- 3 Web-services description language (started: OWL-S)
- 4 Tools for reading/publishing Semantic Web documents (started: Jena, Haystack, Protege)
- 5 Query language for data represented by RDF (started: SPARQL)

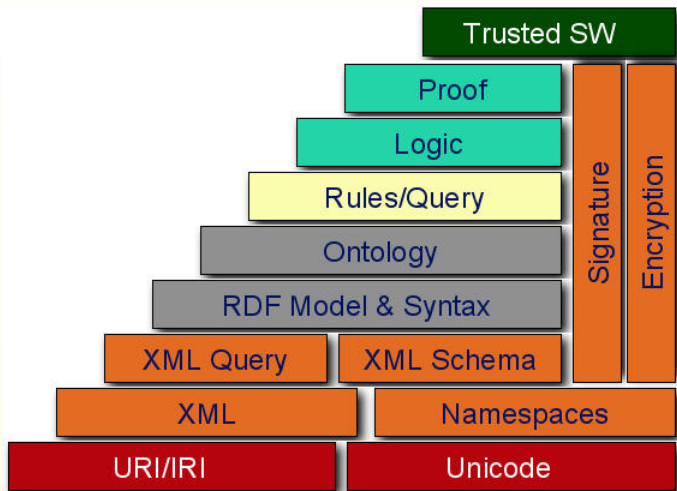
Semantic Web Step-by-Step

- 1 Syntax for knowledge representation (done: RDF)
- 2 Ontology description language (done: OWL)
- 3 Web-services description language (started: OWL-S)
- 4 Tools for reading/publishing Semantic Web documents (started: Jena, Haystack, Protege)
- 5 Query language for data represented by RDF (started: SPARQL)
- 6 Logic reasoning about RDF statements (to be done)

Semantic Web Step-by-Step

- 1 Syntax for knowledge representation (done: RDF)
- 2 Ontology description language (done: OWL)
- 3 Web-services description language (started: OWL-S)
- 4 Tools for reading/publishing Semantic Web documents (started: Jena, Haystack, Protege)
- 5 Query language for data represented by RDF (started: SPARQL)
- 6 Logic reasoning about RDF statements (to be done)
- 7 Semantic search and semantic agents (to be done)

Cake of Tim Berners-Lee



Concept of Semantic Search

What is **semantic search**?

Concept of Semantic Search

What is **semantic search**?

- Assistance to classical web search

Concept of Semantic Search

What is **semantic search**?

- Assistance to classical web search
- Question answering systems

Concept of Semantic Search

What is **semantic search**?

- Assistance to classical web search
- Question answering systems
- Queries that returns concepts (nodes in XML documents), not documents themselves

Concept of Semantic Search

What is **sematic search**?

- Assistance to classical web search
- Question answering systems
- Queries that returns concepts (nodes in XML documents), not documents themselves
- Query is a complex concept (small XML tree), semantic search returns the most similar object

Concept of Semantic Search

What is **semantic search**?

- Assistance to classical web search
- Question answering systems
- Queries that returns concepts (nodes in XML documents), not documents themselves
- Query is a complex concept (small XML tree), semantic search returns the most similar object
- SQL-like queries to database of RDF statements

Concept of Semantic Search

What is **semantic search**?

- Assistance to classical web search
- Question answering systems
- Queries that returns concepts (nodes in XML documents), not documents themselves
- Query is a complex concept (small XML tree), semantic search returns the most similar object
- SQL-like queries to database of RDF statements
- Automated logical inference for RDF statements

Part III

Three Algorithms for Semantic Search

Finding the most specific answer

Concept matching

Identifying related nodes in XML documents

XRANK: Model

Database is a set of **XML documents**

There are **hyperlinks** between nodes

Every node contain some **text**

Query is a short list of keywords

XRANK: Model

Database is a set of **XML documents**

There are **hyperlinks** between nodes

Every node contain some **text**

Query is a short list of keywords

A **complete** answer is a node that together with its descendants contain all query terms

Minimal Answers

A node v is called to be a **minimal answer** if

$$\begin{aligned} & \forall k \in Q : \\ & [v \text{ contains } k] \\ & \text{OR} \\ & [\exists u \text{ son of } v \text{ s.t. } u \text{ contains}^* k \\ & \text{AND } u \text{ is not complete answer}] \end{aligned}$$

Minimal Answers

A node v is called to be a **minimal answer** if

$$\begin{aligned} & \forall k \in Q : \\ & [v \text{ contains } k] \\ & \text{OR} \\ & [\exists u \text{ son of } v \text{ s.t. } u \text{ contains}^* k \\ & \text{AND } u \text{ is not complete answer}] \end{aligned}$$

Search task: find all minimal answers and rank them accordingly to the link/containement popularity

Dewey Code

Nodes in database have Dewey codes $n_1.n_2.\dots.n_h$

For example, Dewey code **7.2.12** denotes the 12th left son of the 2nd left son of the root of the 7th document in our collection.

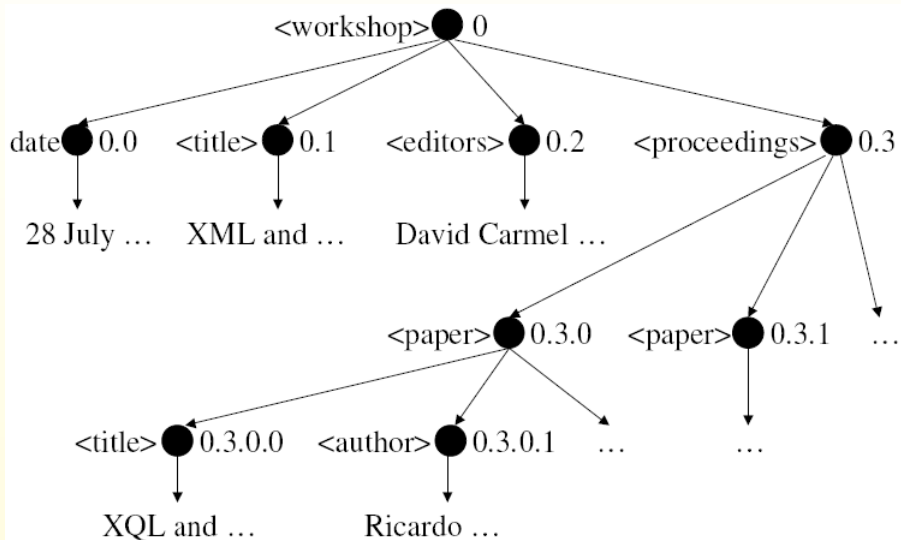
Dewey Code

Nodes in database have Dewey codes $n_1.n_2.\dots.n_h$

For example, Dewey code **7.2.12** denotes the 12th left son of the 2nd left son of the root of the 7th document in our collection.

For every keyword **Dewey inverted index** store a list of Dewey codes of nodes (DIL) that directly contain this keyword

Illustration from XRANK paper



Minimal Answers Problem

Given Dewey inverted lists for all query terms to return a list of Dewey codes of all minimal answers

Algorithm for Minimal Answers (1/2)

Single pass: every time read
a next code in union of DILs

Algorithm for Minimal Answers (1/2)

Single pass: every time read
a next code in union of DILs

Keep an auxiliary data structure **Dewey stack**
for the last scanned read node v :

- for every predecessor of v

- keep a set of keywords

- that are contained* prior-or-equal to v

Algorithm for Minimal Answers (1/2)

Single pass: every time read
a next code in union of DILs

Keep an auxiliary data structure **Dewey stack**
for the last scanned read node v :

- for every predecessor of v
- keep a set of keywords
- that are contained* prior-or-equal to v
- ignoring complete nodes

Algorithm for Minimal Answers (2/2)

Update for Dewey stack from v to u :

- 1 find a lowest common predecessor w for v and u
- 2 Sequentially consider ancestors of u from bottom to top, add keywords of u to their set in Dewey stack
- 3 Stop at root, or with identical set update or on the first complete node
- 4 In latter case output this node to the list of minimal answers

Conceptual Graph Matching

Query is a tree with labelled edges and nodes

Database is a family of trees

Domain information: similarity
between edge/node labels

Conceptual Graph Matching

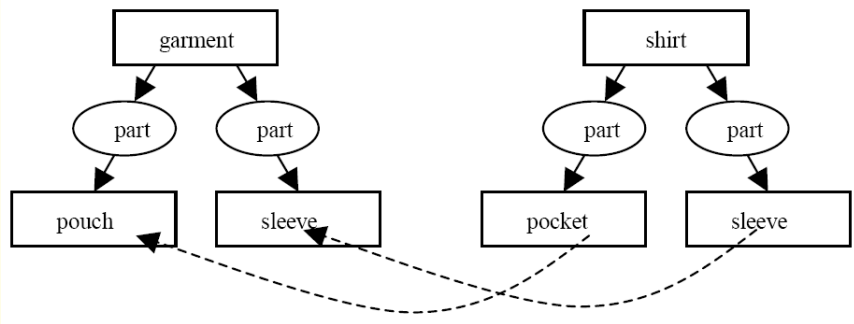
Query is a tree with labelled edges and nodes

Database is a family of trees

Domain information: similarity
between edge/node labels

Task: to find a tree in DB
with maximal similarity to query tree

Illustration from Conceptual Matching Paper



Similarity Formula

$$\begin{aligned} \text{TreeSim}(Q, R) = & \text{NodeSim}(q_0, r_0) + \\ & + \max_{\text{children matching } \pi} \left(\sum_i \text{EdgeSim}(q_0 q_i, r_0 r_{\pi_i}) \cdot \text{TreeSim}(Q|_{q_i}, R|_{r_{\pi_i}}) \right) \end{aligned}$$

Recursive Algorithm for Graph Matching

Compare query tree with every tree in DB separately:

- 1 Compute *TreeSim* for every pair of Q and R roots' children
- 2 Find the best matching by applying Bellman-Ford algorithm

Recursive Algorithm for Graph Matching

Compare query tree with every tree in DB separately:

- 1 Compute *TreeSim* for every pair of *Q* and *R* roots' children
- 2 Find the best matching by applying Bellman-Ford algorithm

Complexity for *l*-branch trees of depth *d*:

$$C(d + 1) = l^2 C(d) + l^4 + \text{const}$$

$$C(d) = \mathcal{O}(l^{2d+2}) = \mathcal{O}(n^2 l^2)$$

Recursive Algorithm for Graph Matching

Compare query tree with every tree in DB separately:

- 1 Compute *TreeSim* for every pair of Q and R roots' children
- 2 Find the best matching by applying Bellman-Ford algorithm

Complexity for l -branch trees of depth d :

$$C(d + 1) = l^2 C(d) + l^4 + \text{const}$$

$$C(d) = \mathcal{O}(l^{2d+2}) = \mathcal{O}(n^2 l^2)$$

In general, time complexity is $\mathcal{O}(n^4)$

XSEarch Model

Database: huge XML tree with labels on internal nodes and keywords on leafs

Query terms: “label:keyword”, “label:”, “:keyword”

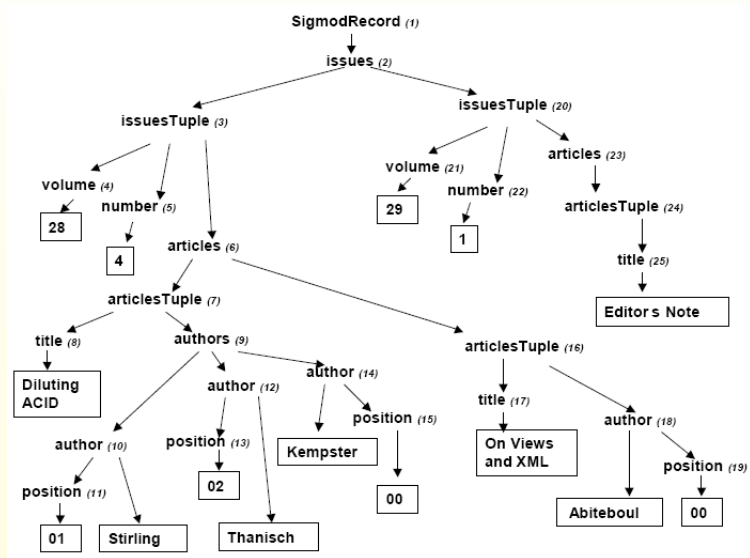
XSEarch Model

Database: huge XML tree with labels on internal nodes and keywords on leafs

Query terms: “label:keyword”, “label:”, “:keyword”

Answer: a set of **interconnected** nodes that together satisfy all query terms

Illustration from XSearch Paper



Interconnection

Nodes u and v are **interconnected** iff on the shortest path between them only labels of u and v can coincide

Properties of Interconnection

For u being ancestor of v :

$$\begin{aligned} InCon[u, v] = & InCon[u, parent(v)] \& \\ & (label(u) \neq label(parent(v))) \& InCon[son_v(u), v] \& \\ & (label(son_v(u)) \neq label(v)) \end{aligned}$$

Properties of Interconnection

For u being ancestor of v :

$$\begin{aligned} InCon[u, v] = & InCon[u, parent(v)] \& \\ & (label(u) \neq label(parent(v))) \& InCon[son_v(u), v] \& \\ & (label(son_v(u)) \neq label(v)) \end{aligned}$$

Otherwise:

$$\begin{aligned} InCon[u, v] = & InCon[u, parent(v)] \& (label(u) \neq \\ & label(parent(v))) \& InCon[parent(u), v] \& \\ & (label(parent(u)) \neq label(v)) \end{aligned}$$

Properties of Interconnection

For u being ancestor of v :

$$\begin{aligned} InCon[u, v] = & InCon[u, parent(v)] \& \\ & (label(u) \neq label(parent(v))) \& InCon[son_v(u), v] \& \\ & (label(son_v(u)) \neq label(v)) \end{aligned}$$

Otherwise:

$$\begin{aligned} InCon[u, v] = & InCon[u, parent(v)] \& (label(u) \neq \\ & label(parent(v))) \& InCon[parent(u), v] \& \\ & (label(parent(u)) \neq label(v)) \end{aligned}$$

Using these formulas we can compute $InCon$ for all pairs in $\mathcal{O}(|T|)$ time by dynamic programming

Directions for Further Research

- Algorithms for **online** conceptual graph matching
- Queries using arithmetic: “what is the most popular movie (according to IMDB) I have not seen yet?”
- Automated inference for RDF statements?
Semantic search for the case when the answer is not in the DB, but can be derived from it.

Highlights

- XRANK: merging Dewey inverted lists by a single pass
- Concept matching: finding the most similar tree to the query tree
- XSEarch: computing interconnection by dynamic programming

Highlights

- XRANK: merging Dewey inverted lists by a single pass
- Concept matching: finding the most similar tree to the query tree
- XSEarch: computing interconnection by dynamic programming

Thanks for participating in this course!

References (1/2)

Course homepage

<http://yury.name/algoweb.html>



L.Guo, F.Shao, C.Botev, J.Shanmugasundaram

XRANK: Ranked Keyword Search over XML Documents

<http://www.cs.fiu.edu/~vagelis/classes/COP6727/publications/XRank.pdf>



S.Cohen, J.Mamou, Y.Kanza, Y.Sagiv

XSEarch: A Semantic Search Engine for XML

<http://wwwdb.informatik.uni-rostock.de/Archiv/vldb2003/papers/S03P02.pdf>



J.Zhong, H.Zhu, J.Li, Y.Yu

Conceptual Graph Matching for Semantic Search

<http://apex.sjtu.edu.cn/docs/iccs2002.pdf>

References (2/2)



R.Guha, R.McCool, E.Miller

Semantic Search

<http://learning.ncsa.uiuc.edu/lmarini/papers/p700-guha.pdf>



S.Harris

SPARQL query processing with conventional relational database systems

<http://eprints.ecs.soton.ac.uk/11126/01/harris-ssws05.pdf>



E.Brill, S.Dumais, M.Banko

An Analysis of the AskMSR Question-Answering System

<http://www.stanford.edu/class/linguist180/EMNLP2002.pdf>



T.Berners-Lee, J.Hendler, O.Lassila

Semantic Web

http://wireless.ictp.trieste.it/school_2002/lectures/canessa/0501berners-lee.ps