INFO/CS 4302
Web Information Systems

FT 2012
Lecture 17: SKOS, SPARQL

- Bernhard Haslhofer -
Plan for today...

- Context & Recap
- SKOS
- SPARQL
- Questions
CONTEXT & RECAP
SPARQL

- A language for describing knowledge organization systems (taxonomies, thesauri, classification schemes)
SPARQL

• A query language and protocol for accessing RDF data on the Web

```sparql
SELECT DISTINCT ?x
WHERE {
  ?x dcterms:subject
}
```
SKOS
Knowledge Organization Systems (KOS)
Folksonomy Examples

All time most popular tags

animals architecture art asia australia autumn baby band barcelona beach berlin bike bird birds birthday black blackandwhite blue bw california canada cannon car cat chicago china christmas church city clouds club color concert dance day de dog england europe fall family fashion festival film florida flower flowers food football france friends fun garden geotagged germany girl girls graffiti green halloween hawaii holiday house india iphone island itsie italy japan kids la lake landscape light live london love macro me mexico model mountain museum music nature new newyork newyorkcity night nikon nyc ocean old paris park party people photography photos portrait raw red river rock san sanfrancisco scotland sea seattle show sky snow spain spring square street summer sun sunset taiwan texas thailand tokyo toronto travel tree trees trip uk urban usa vacation vintage washington water wedding white winter woman yellow zoo

What are tags?
You can give your photos and videos a "tag", which is like a keyword or category label. Tags help you find photos and videos which have something in common. You can assign up to 75 tags to each photo or video.
Introduction

In February 1999, the statistical agencies of Canada, Mexico, and the United States launched a joint multi-phase initiative to develop a comprehensive demand-oriented product classification, known as the North American Product Classification System (NAPCS). Work to date has focused on the products produced by service industries in 12 NAICS sectors 48-49 through 81. With that work nearing completion, this web page provides an overview of and progress report on the NAPCS initiative and presents the final versions of the product lists developed so far for the service industries included in those 12 sectors.

Overview and Progress Report

Final NAPCS Product Lists

- Description of product lists
- View PDF files of individual product lists
- Download sequential Excel file of all product lists

Word to search for:  
Search WordNet

Display Options:  
(Select option to change)  
Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Noun

- **S: (n) cat, true cat** (feline mammal usually having thick soft fur and no ability to roar: domestic cats; wildcats)
- **S: (n) guy, cat, hombre, bozo** (an informal term for a youth or man) "a nice guy"; "the guy's only doing it for some doll"
- **S: (n) cat** (a spiteful woman gossip) "what a cat she is!"
- **S: (n) kat, khat, kat, quatt, cat, Arabian tea, African tea** (the leaves of the shrub Catha edulis which are chewed like tobacco or used to make tea; has the effect of a euphoric stimulant) "in Yemen kat is used daily by 85% of adults"
- **S: (n) cat-o'-nine-tails, cat** (a whip with nine knotted cords) "British sailors feared the cat"
- **S: (n) Caterpillar, cat** (a large tracked vehicle that is propelled by two endless metal belts; frequently used for moving earth in construction and farm work)
- **S: (n) big cat, cat** (any of several large cats typically able to roar and living in the wild)
- **S: (n) computerized tomography, computed tomography, CT, computerized axial tomography, computed axial tomography, CAT** (a method of examining body organs by scanning them with X rays and using a computer to construct a series of cross-sectional scans along a single axis)

Verb

- **S: (v) cat** (beat with a cat-o'-nine-tails)
- **S: (v) vomit, vomit up, purge, cast, sick, cat, be sick, disgorge, regorge, retch, puke, barf, spew, spue, chuck, upchuck, honk, regurgitate, throw up** (eject the contents of the stomach through the mouth) "After drinking too much, the students vomited"; "He purged continuously"; "The patient regurgitated the food we gave him last night"
What is SKOS?

- A model for expressing the basic structure and content of concept schemes such as thesauri, classification schemes, taxonomies, folksonomies, and other similar types of controlled vocabularies
- Allows concepts to be composed and published as Linked Data on the Web
- Hides the complexity of OWL - easy to use
SKOS Concepts are...

- ... identified by URIs
- ... labeled with 1..* natural language strings
- ... documented with various types of notes
- ... semantically linked to each other
- ... aggregated into concept schemes
Example SKOS Concept

Parody films

From Library of Congress Subject Headings

**Details**

**Parody films**

This heading is used as a topical heading for works about films that comically imitate another work or group of works of a more serious nature. When used as a topical heading it is subdivided by the appropriate geographic, topical, and/or form subdivisions.

General works about the use of parody in motion pictures are entered under Parody in motion pictures.

**URI**

<http://id.loc.gov/authorities/sh2001000475#concept>

**Type**

Topical Term

**Alternate Labels**

- Film genre parodies
- Film parodies
- Genre parodies (Motion pictures)
- Genre parody films
- Motion picture parodies
skos:Concept

- Concepts are
  - the **units of thought**: ideas, meanings, categories of objects, etc.
  - abstract entities which are independent of the terms used to label them
skos:(pref|alt|hidden)Label

- Labels refer to concepts’ natural language(s)
  - `skos:prefLabel`: the preferred lexical label
  - `skos:altLabel`: alternative lexical labels (e.g., synonyms)
  - `skos:hiddenLabel`: labels useful for indexing (e.g., typos)

![Diagram]

```
lcsh:sh2007025344#concept
  -- `skos:altLabel`
  "Movie parodies"@en

lcsh:sh2007025344#concept
  -- `skos:altLabel`
  "Send-up films"@en

lcsh:sh2007025344#concept
  -- `skos:prefLabel`
  "Parody films"@en
```
SKOS Semantic Relationships

• The meaning of a concept is also defined by its links to other concepts
  – skos:broader: hierarchical link to a more general concept
  – skos:narrower: hierarchical link to a more specific concept
  – skos:related: associative (non-hierarchical) link
SKOS Documentary Notes

• Add further human-readable documentation
  – skos:scopeNote: info about intended meaning
  – skos:definition: complete explanation of meaning
  – skos:example: example concept use

"Comedy films"@en

“This heading is used as a genre/form heading for films that comically imitate another work or group of works of a more serious nature.”@en
skos:ConceptScheme

- Allow the organization of skos:Concepts in some Knowledge Organization Scheme (KOS)

```
skos:ConceptScheme
lcsh:#genreFormTerms

skos:inScheme

skos:Concept
lcsh:sh2007025038#concept

skos:Concept
...
A Real-World Example

• NY Times topics: http://topics.nytimes.com/

• Dereference and analyze “Jack Nicholson” @ New York Times
  – http://data.nytimes.com
SPARQL
What is SPARQL?

• SPARQL is a query language for accessing RDF data
• SPARQL is a protocol that defines how queries and results can be transported over a network
Example Dataset

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  
@prefix dbpedia: <http://dbpedia.org/resource/>  
@prefix yago: <http://dbpedia.org/class/yago/>  
@prefix foaf: <http://xmlns.com/foaf/0.1/>  
@prefix dbpedia-owl: <http://dbpedia.org/ontology/>  

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix yago: <http://dbpedia.org/class/yago/>  
@prefix foaf: <http://xmlns.com/foaf/0.1/>  
@prefix dbpedia-owl: <http://dbpedia.org/ontology/>  

Jack Nicholson

yago: AmericanFilmActors

dbpedia: Jack_Nicholson

1937-04-22

dbpedia-owl: birthDate

Whoopi Goldberg

dbpedia: Whoopi_Goldberg

1955-11-13

dbpedia-owl: birthDate
```

foaf:name

dbpedia-owl: birthDate
A Simple SPARQL Query

```
SELECT ?name
WHERE {
    dbpedia:Jack_Nicholson foaf:name ?name
}
```

<table>
<thead>
<tr>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Jack Nicholson”</td>
</tr>
</tbody>
</table>
Simple Query Illustrated

- In SPARQL we formulate triple/graph patterns

```sparql
SELECT ?name
WHERE {
  dbpedia:Jack_Nicholson foaf:name ?name
}
```
Simple Query Illustrated

- Patterns are matched against the dataset
Querying Multiple Values

SELECT ?name ?birthdate
WHERE {
  ?x rdf:type yago:AmericanFilmActors .
  ?x foaf:name ?name .
  ?x dbpedia-owl:birthDate ?birthdate .
}

<table>
<thead>
<tr>
<th>name</th>
<th>birthdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Jack Nicholson”</td>
<td>1937-04-22</td>
</tr>
<tr>
<td>“Whoopi Goldberg”</td>
<td>1955-11-13</td>
</tr>
</tbody>
</table>
SELECT ?name ?birthdate
WHERE {
  ?x rdf:type yago:AmericanFilmActors .
  ?x foaf:name ?name .
  ?x dbpedia-owl:birthDate ?birthdate .
}

Exercise: draw the graph pattern
Prefixes in SPARQL queries

- The DBPedia SPARQL Web interface knows about prefix/namespace mappings
- But in general SPARQL endpoints don’t

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbpedia: <http://dbpedia.org/resource/>

SELECT ?name
WHERE {
    dbpedia:Jack_Nicholson foaf:name ?name
}
SPARQL Filters

• ...restrict the solutions of a graph pattern match according to a given expression
• ...eliminate solutions that, when substituted into the expression, result in boolean false

```sparql
SELECT ?name
WHERE {
  ?x foaf:name ?name .
  FILTER regex(?name, "Nicholson")
}
```

<table>
<thead>
<tr>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Jack Nicholson”</td>
</tr>
</tbody>
</table>
SPARQL Filters Example

```sparql
SELECT ?name ?birthdate
WHERE {
  ?x rdf:type yago:AmericanFilmActors .
  ?x foaf:name ?name .
  ?x dbpedia-owl:birthDate ?birthdate .
  FILTER(?birthdate > "1950-01-01T00:00:00Z"^^xsd:dateTime)
}
```

<table>
<thead>
<tr>
<th>name</th>
<th>birthdate</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Whoopi Goldberg”</td>
<td>1955-11-13</td>
</tr>
</tbody>
</table>
More SPARQL Filters

- SPARQL offers lots of FILTERing possibilities
  - Boolean operators  !  ||  &&
  - Comparison operators  =  !=  >  <  >=  <=
  - Arithmetic operators  *  /  -  +
  - RDF element operators  bound(), isURI(), LANG(), STR()
Optional Graph Patterns

• In a standard SPARQL query the entire graph pattern must match in order to retrieve a result
• But we cannot always assume complete structures
  – not all actors have birthdays
  – not all movies have actors
• SPARQL allows to formulate queries that include information in the solution if it is available, but does not reject the solution if parts are missing
Optional Graph Patterns

dbpedia: The_Two_Jakes
  dbpedia-owl: director
  dbpedia: Jack_Nicholson
    rdf:type
    dbpedia-owl: birthDate
  foaf:name
  1937-04-22

yago: AmericanFilmActors
  rdf:type
  dbpedia: Jack_Nicholson
  foaf:name

dbpedia: Whoopi_Goldberg
  rdf:type
  dbpedia-owl: birthDate
  1955-11-13

Whoopi Goldberg
  1955-11-13
  1955-11-13

Jack Nicholson
  1937-04-22
  1937-04-22
Optional Graph Patterns

```
SELECT ?name ?directed_movie
WHERE {
  ?x rdf:type yago:AmericanFilmActors .
  ?x foaf:name ?name .
  ?directed_movie dbpedia-owl:director ?x .
}
```

<table>
<thead>
<tr>
<th>name</th>
<th>directed_movie</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Jack Nicholson”</td>
<td><code>dbpedia:The_Two_Jakes</code></td>
</tr>
</tbody>
</table>

Exercise: we lost Whoopi. Why?
## Optional Graph Patterns

```sparql
SELECT ?name ?directed_movie
WHERE {
  ?x rdf:type yago:AmericanFilmActors .
  ?x foaf:name ?name .
  OPTIONAL {
    ?directed_movie dbpedia-owl:director ?x .
  }
}
```

<table>
<thead>
<tr>
<th>name</th>
<th>directed_movie</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Jack Nicholson”</td>
<td>dbpedia:The_Two_Jakes</td>
</tr>
<tr>
<td>“Whoopi Goldberg”</td>
<td></td>
</tr>
</tbody>
</table>
SPARQL UNION

• Sometimes you want to express „or“ in a graph pattern
• „The graph should match this OR that pattern“
• With the UNION keyword we can define alternative matching graph patterns
SPARQL UNION Example

```
SELECT  ?name  ?directed_movie
WHERE  {

  {?x rdf:type yago:AmericanFilmActors }  
  UNION  
  {?x rdf:type yago:GermanFilmActors }  

  ?x foaf:name  ?name .

  OPTIONAL  {  
    ?directed_movie dbpedia-owl:director  ?x .  
  }
}
```
SPARQL Solution Modifiers

• The results returned by a query are by default unordered

• SPARQL defines the following solution modifiers
  – ORDER BY - reorder the solution sequence
  – DISTINCT - avoid duplicate solutions
  – OFFSET - start after a certain number of solutions
  – LIMIT - limit the output to a number of solutions
SPARQL Solution Modifiers

SELECT ?name
WHERE {
    ?x a foaf:Person .
    ?x foaf:name ?name .
}
LIMIT 100

SELECT ?person (count(DISTINCT ?spouse) as ?spouses)
where {
    ?person a yago:AmericanFilmActors .
}
ORDER BY DESC(?spouses)
LIMIT 100
Further SPARQL features

• We only covered SELECT
• There are also the CONSTRUCT, ASK, and DESCRIBE query forms
• The other forms are useful in practice!
• Please read the SPARQL Primer: http://www.w3.org/TR/rdf-sparql-query/
What is SPARQL?

- SPARQL is a **query language** for accessing RDF data
- SPARQL is a **protocol** that defines how queries and results can be transported over a network
SPARQL Protocol for RDF

- SPARQL also defines how **queries** and **results** can be transported over a network
- Bindings for **HTTP**
  
  [http://www.w3.org/TR/rdf-sparql-protocol/](http://www.w3.org/TR/rdf-sparql-protocol/)
SPARQL Protocol - HTTP

• Queries are sent to an endpoint using
  – HTTP GET (default)
  – HTTP POST (if encoded query string exceed limits)

• Results are returned either as
  – SPARQL Results Document (SELECT and ASK)
  – Serialized RDF Graph (CONSTRUCT and DESCRIBE)
SPARQL Protocol - Example

```
SELECT ?name
WHERE {
    dbpedia:Jack_Nicholson foaf:name ?name
}
```

**URI-encoded query string (EncodedQuery)**

```
SELECT%20?name%0AWHERE%20%7B%0A%20%20%20%20dbpedia:Jack_Nicholson%20foaf:name%20?name%0A%7D
```
SPARQL Protocol - Example

GET /sparql/?query=EncodedQuery HTTP/1.1
Host: dbpedia.org

Using CURL:

curl -v http://dbpedia.org/sparql/?query=SELECT%20?name%0AWHERE%20%7B%0A%20%20%20dbpedia:Jack_Nicholson%0A%20%20%20foaf:name%20?name%0A%7D
SPARQL Protocol - Example

<sparql xmlns="...">
<head>
  <variable name="name"/>
</head>
<results distinct="false" ordered="true">
  <result>
    <binding name="name">
      <literal xml:lang="en">
        Jack Nicholson
      </literal>
    </binding>
  </result>
</results>
QUESTIONS?