Creating Semantic Mashups:

Bridging Web 2.0 and the Semantic Web

Jamie Taylor, Colin Evans, Toby Segaran

freebase™
Why is Semantic Data Interesting?
Why is Semantic Data Interesting?

• Walmart demo
Why is Semantic Data Interesting?

- Walmart demo
  - http://blog.kiwitobes.com/?p=51
Why is Semantic Data Interesting?

• Walmart demo
  • http://blog.kiwitobes.com/?p=51

• Political Query
Why is Semantic Data Interesting?

- Walmart demo
  - http://blog.kiwitobes.com/?p=51
- Political Query
  - http://www.freebase.com/view/guid/9202a8c04000641f8000000008053940
Why is Semantic Data Interesting?

• Walmart demo
  • http://blog.kiwitobes.com/?p=51

• Political Query
  • http://www.freebase.com/view/guid/9202a8c04000641f8000000008053940

• Venture Spin
Why is Semantic Data Interesting?

• Walmart demo
  • http://blog.kiwitobes.com/?p=51

• Political Query
  • http://www.freebase.com/view/guid/9202a8c04000641f8000000008053940

• Venture Spin
  • http://www.perlgoddess.com/FreeSpin/FreeSpin.swf
Semantic Data is Flexible Data

- The data for these demos all used structured semantics
- The data was not specifically designed for the demo
- The demos can utilize any data set with shared semantics (e.g., Venture Spin)
Overview

• Introduction to semantic ideas
Overview

• Introduction to semantic ideas
• Technologies and Architectural techniques
Overview

• Introduction to semantic ideas
• Technologies and Architectural techniques
• Build something now looking to the Future
Goals
Goals

• Enough to get you started with semantic technologies
Goals

• Enough to get you started with semantic technologies

• Understand advantages and issues with semantic architectures
Goals

• Enough to get you started with semantic technologies

• Understand advantages and issues with semantic architectures

• Basic understanding of semantic representation
Goals

• Enough to get you started with semantic technologies

• Understand advantages and issues with semantic architectures

• Basic understanding of semantic representation

• Ability to use basic semantic repository
Goals

• Enough to get you started with semantic technologies

• Understand advantages and issues with semantic architectures

• Basic understanding of semantic representation

• Ability to use basic semantic repository

• Working overview of a semantic system
As web developers we want to:

• Increase the utility of our applications
  e.g., help users get stuff done

• Build applications with greater efficiency
Web 1.0

- Single function applications
- Publishing large private databases
Web 1.0: Stovepipes

Diner and a Movie

(citysearch.com)

MAPQUEST.

(moviefone)
Web 1.0: Stovepipes

- Data is in silos
- No information sharing except in the user’s head
- The end user drives system and data integration
  ...usually through “copy & paste”
Web 2.0

- Leverage silos of content
- User-generated content
- Open APIs facilitate mash-ups
- The “Social Web”
Web 2.0: UI Mashups
Web 2.0: UI Mashups

- Mash-ups only allow shallow integration at the UI
- Data is still in silos
- User-generated content is also in silos

Data doesn't stray far from its point of creation
• Even with open APIs and mash-ups, users still do most of the system integration

• With the proliferation of user-generated content, system integration is more important than ever!

• Data, whether user-generated, or proprietary, is not easily accessible or transferable

• We’re still fighting with stovepipe systems
History of Web Integration

Users’ Brain
Web 1.0

Point of Integration
History of Web Integration

Users’ Brain
Web 1.0

UI (Mash-up)
Web 2.0

Point of Integration
History of Web Integration

- Users’ Brain
- Web 1.0
- UI (Mash-up)
- Web 2.0
- Semantic Mash-ups

Point of Integration
Integration Scaling

Web 2.0 Mashup

Utility increase as number of sources increases

Users benefit as more data is made available in application
Web 2.0 Mashup

Integration effort grows with number of sources.

Easy to integrate first few sources, but complexity increases as number of sources increases.
Integration Scaling

Semantic Mashup

Treat sources uniformly

Pay a slightly higher start-up cost, but quickly benefit.

Note: red line is should somewhat sloping up :-)

# Sources
Why Semantics

• Developing Content is expensive
Why Semantics

• Developing Content is expensive
• Developing Web applications is expensive
Why Semantics

• Developing Content is expensive
• Developing Web applications is expensive
• Use existing systems/sources where possible
Cracking the Stovepipe

• Semantics facilitate shared meaning through
  • Subject Identity
  • Strong Semantics
  • Open APIs + Open Data

• These principles make it much easier to combine stovepipe systems and integrate data
Ridley Scott directed Blade Runner
Creating Meaning

Ridley Scott directed Blade Runner

subject
Creating Meaning

Ridley Scott  directed  Blade Runner

subject  directed  predicate
Creating Meaning

Ridley Scott directed Blade Runner

subject

directed

object
Creating Meaning
Creating Meaning

Ridley Scott directed
Creating Meaning

Ridley Scott

directed

Blade Runner
Creating Meaning

Ridley Scott directed Blade Runner.
Using Shared Meaning

Creating Triples in Javascript:

```javascript
myRDF = new RDF()
t1 = new Triple('A', 'geo', '37.44, -122.14')
t2 = new Triple('B', 'company', 'Wal-mart')
myRDF.addTriples([t1, t2])
```

http://rdflib.net/
Using Shared Meaning

Installed Services

Google GEO Service: address -> geo coordinates
Freebase business chains: company -> addresses
Upcoming event search: event -> addresses
Freebase industry search: industry -> company

http://kiwitobes.com/maptest/
Using Shared Meaning

Example of a service (Freebase):

```javascript
function businessindustry(store) {
    at=store.Match(null,null,'industry',null)
    for (i=0;i<at.length;i++) {
        subject=at[i].subject
        industry=at[i].object

        query=[{"type":"/business/company",
                'name':null,
                'industry':industry}]

        Metaweb.read(query,
                     function(r) {
                         t=[]
                         for (i=0;i<r.length;i++) {
                             t.push(new Triple(subject,
                                               'company',r[i].name,'','','en'))
                         }
                         store.addTriples(t)
                     })
    }
}
```
Example of a service (Upcoming):

```javascript
function eventsearch(store) {
    at=store.Match(null,null,'event',null)
    for (i=0;i<at.length;i++) {
        subject=at[i].subject
        event=at[i].object

        var request = new XMLHttpRequest();
        request.open("GET", 'upcomingread.php?query='+event, true);
        request.onload = function()
        {
            if (request.readyState == 4) {
                var items = request.responseXML.getElementsByTagName("event");
                t=[]
                for (j=0;j<items.length;j++) {
                    address=items[j].getAttribute('venue_address')+', '+
                    items[j].getAttribute('venue_city')+', '+
                    items[j].getAttribute('venue_state_code')+' '+
                    items[j].getAttribute('venue_zip')

                    t.push(new Triple(subject,'address',address))
                }
                store.addTriples(t)
            }
        }
        request.send(null);
    }
}
```
Identifying Shared Meaning
The Meaning of “is” is

http://dbpedia.org/resource/IS
The Meaning of “is” is

http://dbpedia.org/resource/IS

• URI’s provide strong references
The Meaning of “is” is

http://dbpedia.org/resource/IS

• URI’s provide strong references
• Much like pointing in the physical world
The Meaning of “is” is

http://dbpedia.org/resource/IS

• URI’s provide strong references
• Much like pointing in the physical world
  “this is red”
The Meaning of “is” is

http://dbpedia.org/resource/IS

- URI’s provide strong references
- Much like pointing in the physical world
  “this is red”
  “this is a pen”
The Meaning of “is” is

http://dbpedia.org/resource/IS

• URI’s provide strong references

• Much like pointing in the physical world
  “this is red”
  “this is a pen”

• a URIref is an unambiguous pointer to something of meaning
Creating Meaning

http://...
ridley_scott

http://...
blade_runner
Creating Meaning

http://... ridley_scott

http://... blade_runner
Creating Meaning

http://...directed

blade_runner

http://...

http://...directed

http://...

http://...

http://...

ridley_scott
Creating Meaning

http://... directed

blade_runner

http://...

ridley_scott

http://... directed

blade_runner
Creating Meaning

http://...directed

subject

http://...directed

predicate

http://...blade_runner

object
Creating Meaning

```python
fb = Namespace("http://www.freebase.com/view/en/")
graph.add(
    ( fb("blade_runner"),
        fb("directed_by"),
        fb("ridley_scott")
    )
)
```
Two Types of URIrefs
Two Types of URIrefs

• Things/states (subjects, objects)
Two Types of URIrefs

- Things/states (subjects, objects)
  - Blade Runner
Two Types of URIs

• Things/states (subjects, objects)
  • Blade Runner
  • Ridley Scott
Two Types of URIrefs

- Things/states (subjects, objects)
  - Blade Runner
  - Ridley Scott
  - Movies
Two Types of URIrefs

- Things/states (subjects, objects)
  - Blade Runner
  - Ridley Scott
  - Movies
- Relations (predicates)
Two Types of URIrefs

- Things/states (subjects, objects)
  - Blade Runner
  - Ridley Scott
  - Movies
- Relations (predicates)
  - directed by
Two Types of URIrefs

- Things/states (subjects, objects)
  - Blade Runner
  - Ridley Scott
  - Movies

- Relations (predicates)
  - directed by
  - acted in
Graph Data Models
Graph Data Models

"Blade Runner"

name
Graph Data Models

"Blade Runner"

- name
- release date

Jun 25, 1982
Graph Data Models

"Blade Runner"

name

release date

1981

actor

"Harrison Ford"

Graph Data Models

"Blade Runner"

name

release date

1981

actor

"Harrison Ford"
Graph Data Models

"Blade Runner"
- name
- release date: Jun 25, 1982

"Harrison Ford"
- name

actor

Graph Data Models

- "Blade Runner" with release date Jun 25, 1982
- "Harrison Ford" with birth date Jul 13, 1942
from rdflib import *

fb = Namespace("http://www.freebase.com/view/en/")
graph = ConjunctiveGraph()

br = fb("blade_runner")
graph.add((br, fb("name"), Literal("Blade Runner")))
graph.add((br, fb("release_date"), Literal("Jun 25, 1982")))

hf = fb("harrison_ford")
graph.add((hf, fb("name"), Literal("Harrison Ford")))
graph.add((hf, fb("birth_date"), Literal("Jul 13, 1942")))

graph.add((br, fb("actor"), hf))
Graph Integration
Graph Integration
Graph Integration
Graph Integration
Graph Integration
W3C Vision

Tim Berners-Lee’s
Giant Global Graph
Stack Attack: J2EE
Take What You Need
Take What You Need
Linked Open Data

- Web of Open Data ("global graph")
- Expressed in RDF
- Lack of ontological agreement
  - how many ways are there to express lat/lon?!
- Canonical references are problematic
- Closest thing we have to the Semantic Web
  ...more like a test bed
Tabulator

Browsing the Global Graph

http://dig.csail.mit.edu/2005/ajar/ajaw/data#Tabulator
Open Data

http://demo.openlibrary.org/dev/docs/data

http://theinfo.org/get/data
Don’t get caught up in the serial representation - any RDF library will take care of that for you transparently.

Focus on the data model
Just Enough RDF

• RDF is a Data Model

Don’t get caught up in the serial representation - any RDF library will take care of that for you transparently.

Focus on the data model
• RDF is a Data Model
• A very simple model!
Just Enough RDF

• RDF is a Data Model
  • A very simple model!
• RDF has many (inconvenient) serializations

Focus on the data model

Don't get caught up in the serial representation - any RDF library will take care of that for you transparently.
Just Enough RDF

• RDF is a Data Model
  • A very simple model!
• RDF has many (inconvenient) serializations
  • RDF-XML

Don't get caught up in the serial representation - any RDF library will take care of that for you transparently.
Focus on the data model
Just Enough RDF

• RDF is a Data Model
  • A very simple model!
• RDF has many (inconvenient) serializations
  • RDF-XML
  • N3

Don’t get caught up in the serial representation - any RDF library will take care of that for you transparently.

Focus on the data model
Just Enough RDF

• RDF is a Data Model
  • A very simple model!
• RDF has many (inconvenient) serializations
  • RDF-XML
  • N3
  • Turtle

Don't get caught up in the serial representation - any RDF library will take care of that for you transparently.

Focus on the data model
RDF Data Model

- Nodes (“Subjects”)
- connect via Links (“Predicates”)
- to Objects
  - either Nodes or Literals
RDF Data Model

- Nodes are referenced by URIs (http://foo/bar/)
- Links are referenced by URIs
- Literals are text strings, sometimes with a URI type and a language attached
- Literal types typically are XML Schema URIs (examples)
• RDF is typically expressed in statements or triples
• Triples are composed of a node, a link, and either another node or a literal
• <http://www.w3.org/People/Berners-Lee/card#i>
  <http://www.w3.org/2000/01/rdf-schema#label>
  “Tim Berners-Lee”
• RDF triples are typically grouped into graphs
• Graph Query
  • Triple \((s, p, o)\)
• Graph query languages (RDQL, SPARQL)
from rdflib import *

fb = Namespace("http://www.freebase.com/view/en/")

graph = ConjunctiveGraph()

starredin = fb["starred_in"]

graph.add((fb["carrie_fisher"], starredin, fb["star_wars"]))
graph.add((fb["harrison_ford"], starredin, fb["star_wars"]))
graph.add((fb["harrison_ford"], starredin, fb["blade_runner"]))
graph.add((fb["daryl_hannah"], starredin, fb["blade_runner"]))
for triple in graph.triples((None, starredin, fb["star_wars"])): print triple

for subject in graph.subjects(predicate=starredin, object=fb["star_wars"]): print subject
SELECT ?costar WHERE {
}
print list(graph.query(

"""SELECT ?costar WHERE {
} """,
initNs=dict(fb=Namespace("http://www.freebase.com/view/en/")))}
μformats

• Semantics embedded in display markup (XHTML)
• Strong (predefined) semantics
  • Each μformat defines an “ontology”

<div class="hreview">
  <span><span class="rating">5</span> out of 5 stars</span>
  <h4 class="summary">Crepes on Cole is awesome</h4>
  <span class="reviewer vcard">Reviewer: <span class="fn">Tantek</span> - <abbr class="dtreviewed" title="20050418T2300-0700">April 18, 2005</abbr></span>
  <div class="description item vcard">
    <p>Crepes on Cole is one of the best little creperies in San Francisco. Excellent food and service. Plenty of tables in a variety of sizes for parties large and small.</p>
    <p>Visit date: <span>April 2005</span></p>
    <p>Food eaten: <span>Florentine crepe</span></p>
  </div>
</div>

WP identifies 22 distinct places called San Francisco in the world
RDFa

• Yet another RDF serialization
• Like μ formats, embeddable in HTML
• Like RDF high expressability + extensibility
• Like any RDF serialization, you don’t want to create them by hand!

<p xmlns:dc="http://purl.org/dc/elements/1.1/"
    about="http://www.example.com/books/wikinomics">In his latest book
<cite property="dc:title">Wikinomics</cite>,
<span property="dc:author">Don Tapscott</span> explains deep changes in technology,
demographics and business.
The book is due to be published in
<span property="dc:date" content="2006-10-01">October 2006</span>.</p>
What I mean by Ontology
What I mean by Ontology

Ontology:
Ontology:

An explicit specification of a conceptualization
Ontology:

An explicit specification of a conceptualization

Conceptualization:
Ontology: An explicit specification of a conceptualization

Conceptualization: Abstract, simplified view of the world that we wish to represent for some purpose
What I mean by Ontology

Ontology:
An explicit specification of a conceptualization

Conceptualization:
Abstract, simplified view of the world that we wish to represent for some purpose
Ontology

IS NOT:
Ontology

IS NOT:

• Magic
Ontology

IS NOT:

• Magic

• Universal
Ontology

IS NOT:

- Magic
- Universal
- Change the world
Ontology

**IS NOT:**
- Magic
- Universal
- Change the world

**IS:**
Ontology

**IS NOT:**

- Magic
- Universal
- Change the world

**IS:**

- An artifact
Ontology

IS NOT:

• Magic
• Universal
• Change the world

IS:

• An artifact
• An API
Ontology

**IS NOT:**
- Magic
- Universal
- Change the world

**IS:**
- An artifact
- An API
- A Social Contract
Movie Ontology

- movie
  - name
  - release_date
  - imdb_rating
  - rt_rating
Movie Ontology

- movie
  - name
  - release_date
  - imdb_rating
  - rt_rating
  - actor

- actor
  - name
Movie Ontology

- movie
  - name
  - release_date
  - imdb_rating
  - rt_rating
  - actor
  - showing

- actor
  - name

- show
  - showing
  - time

- theater
  - name
  - address
from rdflib import *

fbCommon = Namespace("http://www.freebase.com/view/common/")
oName = fbCommon["object/name"]
oType = fbCommon["object/type"]

fbPeople = Namespace("http://www.freebase.com/view/people/")
personType = fbPeople["person"]
pPhoto = fbPeople["person/photo"]

fbFilm = Namespace("http://www.freebase.com/view/film/")
filmType = fbFilm["film"]
fImdbId = fbFilm["film/imdb_id"]
fImdbRating = fbFilm["film/imdb_rating"]
fRtRating = fbFilm["film/rt_rating"]
fActor = fbFilm["film/actor"]

theaterType = fbFilm["theater"]
tAddress = fbFilm["theater/address"]
tShowing = fbFilm["theater/showing"]

showingType = fbFilm["showing"]
sTime = fbFilm["showing/time"]

fbDining = Namespace("http://www.freebase.com/view/dining/")
restaurantType = fbDining["restaurant"]
rAddress = fbDining["restaurant/address"]
What is Freebase?

- Structured Database
- Strong Collaboratively Edited Subjects
- Strong Collaboratively Developed Semantics
- Open API + Open Data
What’s in Freebase?

- Over 3.3 million subjects
- ~750,000 people
- ~450,000 locations
- ~50,000 companies
- ~40,000 movies
- Over 1000 types and 3000 properties
Blade Runner is a 1982 American cyberpunk film, directed by Ridley Scott. The screenplay, written by Hampton Fancher and David Peoples, was based on the novel Do Androids Dream of Electric Sheep? by Philip K. Dick. The film stars Harrison Ford and features Rutger Hauer, Sean Young, Edward James Olmos, M. Emmet Walsh, and Daryl Hannah. The film depicts a dystopia Los Angeles in November 2019 in which genetically manufactured beings called replicant – visually indistinguishable from adult humans – are used for dangerous and degrading work in Earth’s “off-world colonies”. Following a small replicant uprising, replicants become illegal on Earth; and specialist police called “blade runners” are trained to hunt down and “retire” (kill) escaped replicants on Earth. The plot focuses on a brutal and cunning group of replicants hiding in Los Angeles and a semi-retired blade runner, Rick Deckard (Ford), who reluctantly agrees to take on one more assignment. Blade Runner initially polarized.

Read full article at wikipedia.org

Write new description for Freebase.com

Film

Initial release date

Jun 25, 1982

Tagline

Man has made his match ... now it's his problem.
Man Has Made His Match...Now It's His Problem
A chilling, bold, mesmerizing, futuristic detective thriller.
Man has made his match... Now it's his problem.

Directed by

Ridley Scott

Performances

actor

character

Harrison Ford
Rick Deckard
Rutger Hauer
Roy Batty
Sean Young
Rachael
Daryl Hannah
Pris
Joanna Cassidy
Zhora
Brion James
Leon Kowalski
Joe Turkel
Eldon Tyrell
William Sanderson
J.F. Sebastian
Edward James Olmos
Gaff
James Hong
Hannibal Chew

Produced by

Michael Deeley, Bud Yorkin

<table>
<thead>
<tr>
<th>Film</th>
<th>Tagline</th>
</tr>
</thead>
</table>
|      | Man has made his match ... now it's his problem.  
|      | Man Has Made His Match...Now It's His Problem 
|      | A chilling, bold, mesmerizing, futuristic detective thriller. 
|      | Man has made his match... Now it's his problem. |

| Directed by | Ridley Scott |
| Performance | actor character |
| Harrison Ford | Rick Deckard |
| Rutger Hauer | Roy Batty |
| Sean Young | Rachael |
| Daryl Hannah | Pris |
| Joanna Cassidy | Zhora |
| Brian James | Leon Kowalski |
| Joe Turkel | Eldon Tyrell |
| William Sanderson | J.F. Sebastian |
| Edward James Olmos | Gaff |
| James Hong | Hannibal Chew |

| Produced by | Michael Deeley, Bud Yorkin |
| Screenplay by | Phillip K. Dick, Hampton Fancher, David Peoples |
| Story by | Phillip K. Dick |
| Cinematography | Jordan Cronenweth |
| Edited by | Terry Rawlings, Marsha Nakashima, Les Healey |
| Music by | Vangelis |
| Genres | Film noir, Cyberpunk, Science fiction |

<table>
<thead>
<tr>
<th>Runtime</th>
<th>film cut</th>
<th>note</th>
</tr>
</thead>
<tbody>
<tr>
<td>117.0min</td>
<td>Final cut</td>
<td></td>
</tr>
<tr>
<td>115.0min</td>
<td>Director's cut</td>
<td>Director's Cut</td>
</tr>
<tr>
<td>118.0min</td>
<td>International cut</td>
<td>International cut</td>
</tr>
</tbody>
</table>

| Languages | German language, Japanese, Cantonese, English |
| Country of origin | United States |
| Notable filming locations | Ennis House, Bradbury Building |
| Estimated budget | currency amount |
| US $ | 28,000,000.0 |
| Rated | film rating minimum unaccompanied age |
| R (USA) | 17 |
| Film Collections | Dystopian films, AFPI's 100 Years... 100 Movies |
| Soundtrack | Blade Runner |
| Subjects | Identity, Morality, Artificial Intelligence |
Freebase Data Model

Film

Type Key: film

- Included types: Topic
- Also known as: Motion Picture, Movie

User Created Properties

- **Initial release date**
  - Property Key: initial_release_date
  - Expected Type: Date/Time
  - Property Description: no description
    - This will be used in tips and documentation to help others fill in this property.
  - Advanced Options:
    - Do not display in default template
    - Default Orientation:
      - Vertical List
      - Horizontal List
    - Restrict to one value
    - Display as disambiguator

- **Tagline** Text

- **Directed by** Film director (links in as Films directed)

- **Performances** Film performance (links in as Film)
• JSON structure
• Schemas (ontologies) form object abstraction
• Query by example
  Fill in the parts you know
  Result fills in the rest
• JSON structure
• Schemas (ontologies) form object abstraction
• Query by example
  Fill in the parts you know
  Result fills in the rest

Show me the IMDB links for films by George Lucas:

```json
[
  {
    "name" : null,
    "imdb_id" : [ ],
    "initial_release_date":null,
    "directed_by":"George Lucas",
    "type" : "/film/film"
  }
]
```
Carrie Fisher’s Costars:

```json
[
  {
    "film" : [{
      "film" : {
        "name" : null,
        "starring" : [
          {
            "actor" : null
          }
        ],
        "limit" : 2
      },
      "name" : null,
      "starring" : [{
        "actor" : null
      },
      "limit" : 2
    },
    "id" : "/en/carrie_fisher",
    "type" : "/film/actor"
  }
]
A Semantic Architecture

Semantic Architecture

- A little knowledge... goes a long way
- Leverage Silos of Content
- Effort $\propto$ semantic coverage
A Semantic Architecture

Semantic Architecture

Semantic Mapping Layer
A Semantic Architecture

Semantic Architecture

Semantic Mash-up Layer

Semantic Plugin Layer

WWW

WWW

WWW
Film Mashup

- Strong Identity through IMDB IDs
- Pulls data from:
  - IMDB (movie & actor data & rating)
  - Rotten Tomatoes (rating)
  - Freebase (pictures & restaurants)
  - Fandango (movie theaters)
MIT SIMILE

Longwell  Piggy Bank  Solvent  Semantic Bank
Timeline  Timeplot  Exhibit  Babel
Welkin  Crowbar  Referee  Gadget
Appalachian  Banach  HTTPTracer

http://www.cse.msu.edu/~dunham/exhibit/top100.html
Useful Places

• Freebase/MQL:
  • http://www.freebase.com/

• Javascript RDF Library (used in Toby’s map demo)
  • http://www.jibbering.com/rdf-parser/

• LIBLE (Python)
  • http://rdflib.net/

• MIT Semantic Visualization Widgets
  • http://simile.mit.edu/
Useful Places

• SPARQL:
  • http://www.w3.org/TR/rdf-sparql-query/

• Linked Open Data/Semantic Web Interest Group (SWIG)
  • http://www.w3.org/2001/sw/interest/
  • http://www.w3.org/DesignIssues/LinkedData.html

• Tabulator (Linked Open Data Browser):
  • http://www.w3.org/2005/ajar/tab