UNIVERSITY OF JYVÄSKYLÄ

Lecture 6: Data Exchange and Semantic Annotation

TIES4520 Semantic Technologies for Developers
Autumn 2018

University of Jyväskylä

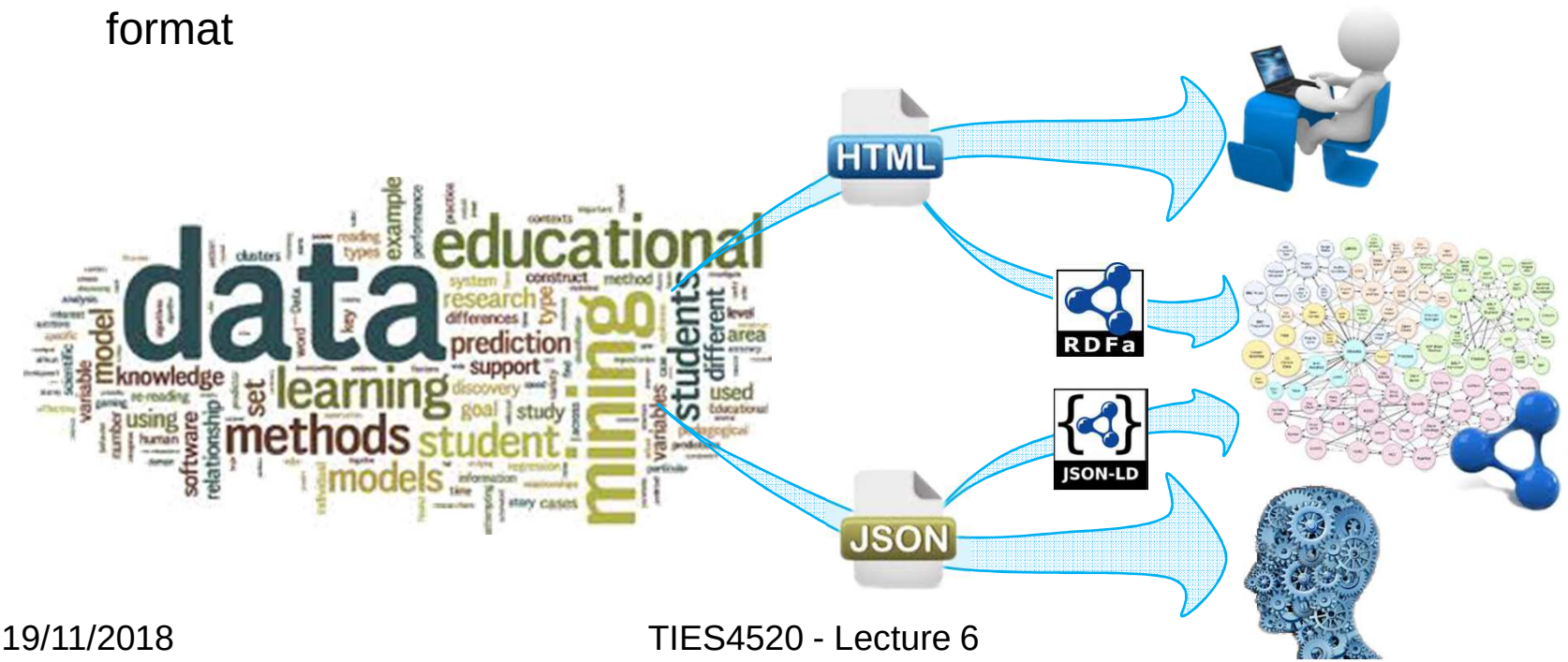
Khriyenko Oleksiy

Part 1

Data Exchange

Machine readable data exchange

- ***RDFa*** – Resource Description Framework in attributes (W3C Recommendation). It is a domain-independent way to explicitly embed RDF data in attributes of a web page to:
  - transfer data from an application to another through the web;
  - write data only once for web users and web applications.
- ***JSON-LD*** - JavaScript Object Notation for Linked Data. Extension of JSON - simple property-value type machine readable data exchange format



RDFa

- ***RDFa 1.1*** is for XHTML and HTML5, also works for any XML-based languages like SVG. (You can use HTML+RDFa, but it won't be officially valid HTML file)
  - ***RDFa Lite 1.1*** - is a minimal subset of RDFa (*http://www.w3.org/TR/rdfa-lite/*)
  - ***RDFa 1.1 Prime*** – is Rich Structured Data Markup for Web Documents (*http://www.w3.org/TR/rdfa-primer/*)
  - ***RDFa Core 1.1*** – is complete specification of RDFa (*http://www.w3.org/TR/rdfa-syntax/*)

- Useful links:
  - Basic presentation: *http://www.slideshare.net/fabien_gandon/rdfa-in-a-nutshell-v1*
  - RDFa materials for users and developers: *http://rdfa.info/* , *http://rdfa.info/dev/*
  - Real-time RDFa 1.1 editor: *http://rdfa.info/play/*
  - RDFa Online Parser: *http://rdf-translator.appspot.com/*
  - RDFa 1.1 Distiller: *http://www.w3.org/2012/pyRdfa/*

RDFa

```
<div vocab="http://schema.org/"
     prefix="ex: http://example.com/"
     resource="ex:alice/posts/trouble_with_bob"
     typeof="Article">
   <h2 property="title">The trouble with Bob</h2>
    ...
   The trouble with Bob is that he takes much better photos than I do:
    ...
   <div resource="ex:bob/photos/sunset.jpg"
        prefix="dc: http://purl.org/dc/terms/" >
      <img src="http://example.com/bob/photos/sunset.jpg" />
      <span property="title">Beautiful Sunset</span>
      by <span property="dc:creator">Bob</span>.
   </div>
</div>
```

The trouble with Bob

…
The trouble with Bob is that he takes much better photos than I do:
…

Beautiful Sunset by Bob

RDFa

```
<div vocab="http://schema.org/"
     prefix="ex: http://example.com/"
     resource="ex:alice/posts/trouble_with_bob"
     typeof="Article">
   <h2 property="title">The trouble with Bob</h2>
    ...
   The trouble with Bob is that he takes much better photos than I do:
    ...
   <div resource="ex:bob/photos/sunset.jpg"
        prefix="dc: http://purl.org/dc/terms/" >
      <img src="http://example.com/bob/photos/sunset.jpg" />
      <span property="title">Beautiful Sunset</span>
      by <span property="dc:creator">Bob</span>.
   </div>
</div>
```

```
@prefix  sc: <http://schema.org/> .
@prefix  ex: <http://example.com/> .
@prefix  dc: < http://purl.org/dc/terms/> .

ex:alice/posts/trouble_with_bob a sc:Article; sc:title "The trouble with Bob".
ex:bob/photos/sunset.jpg sc:title "Beautiful Sunset" ; dc:creator "Bob" .
```

RDFa Lite 1.1

- ***RDFa Lite*** consists of five simple attributes: ***vocab*** (default vocabulary, applied until be redefined), ***typeof***, ***property***, ***resource*** and ***prefix***.


```
<div vocab=“http://schema.org/”
     prefix=“ex: http://example.com/”
     resource="ex:alice/posts/trouble_with_bob“
     typeof=“Article”>
   <h2 property="title">The trouble with Bob</h2>
    ...
   The trouble with Bob is that he takes much better photos than I do:
    ...
   <div resource="ex:bob/photos/sunset.jpg"
        prefix=“dc: http://purl.org/dc/terms/” >
      <img src="http://example.com/bob/photos/sunset.jpg" />
      <span property=“title">Beautiful Sunset</span>
      by <span property=“dc:creator">Bob</span>.
   </div>
</div>
```

- A full list of pre-declared prefixes: *http://www.w3.org/2011/rdfa-context/rdfa-1.1*


```
@prefix  sc: <http://schema.org/> .
@prefix  ex: <http://example.com/> .
@prefix  dc: < http://purl.org/dc/terms/> .

ex:alice/posts/trouble_with_bob a sc:Article; sc:title “The trouble with Bob”.
ex:bob/photos/sunset.jpg sc:title “Beautiful Sunset” ; dc:creator “Bob” .
```

More of RDFa

- It is possible to define a *blank node* by named blank node (e.g. "*_:name*") or just by mentioning a type of the node.
- If the element contains the *href* (or *src*) attribute, property is automatically associated with the value of the attribute rather than the textual content of the <*a*> element.


```
<div vocab="http://xmlns.com/foaf/0.1/">
   <ul>
      <li typeof="Person">
        <a property="homepage" href="http://example.com/bob/">Bob</a>
      </li>
      <li typeof="Person">
        <a property="homepage" href="http://example.com/eve/">Eve</a>
      </li>
   </ul>
</div>
```

```
<div vocab="http://xmlns.com/foaf/0.1/" typeof="Person">
 <span property="name">Alice Birpemswick</span>,
 <ul>
  <li property="knows" typeof="Person">
   <a property="homepage" href="http://example.com/bob/"><span property="name">Bob</span></a>
  </li>
  <li property="knows" typeof="Person">
   <a property="homepage" href="http://example.com/eve/"><span property="name">Eve</span></a>
  </li>
 </ul>
</div>
```

More of RDFa

- HTML+RDFa allows ”*Property copying*” in case you have repeating set of data. It is possible to collect a number of statements as a pattern (***rdfa:Pattern***) and refer to it using the property ***rdfa:copy***.


```
<body vocab="http://purl.org/dc/terms/">
   <div resource="/alice/posts/trouble_with_bob">
      <h2 property="title">The trouble with Bob</h2>
      /some repeating part/
   </div>
   <div resource="/alice/posts/jims_concert">
      <h2 property="title">I was at Jim's concert the other day</h2>
      /some repeating part/
   </div>
</body>
```

```
<body vocab="http://purl.org/dc/terms/">
   <div resource="/alice/posts/trouble_with_bob">
      <h2 property="title">The trouble with Bob</h2>
      <link property="rdfa:copy" href="#cpattern"/>
   </div>
   <div resource="/alice/posts/jims_concert">
      <h2 property="title">I was at Jim's concert the other day</h2>
      <link property="rdfa:copy" href="#cpattern"/>
   </div>
   <div resource="#cpattern" typeof="rdfa:Pattern">
      /some repeating part/
   </div>
</body>
```

More of RDFa

- RDFa allows the value of a ***property*** and ***typeof*** attributes to be a list of values

```
...
<div resource="ex:me" typeof="foaf:Person schema:Person" >
   ...
   <h3 property="dc:creator schema:creator" resource="ex:me">John</h3>
</div>
...
```

- Human readability vs. unambiguity for machine readability (*RDFa Core*)

```
...
<p>Date: <span property="http://purl.org/dc/terms/created">2016-11-28</span></p>   ...
<p>Date: <span property="http://purl.org/dc/terms/created">28th of November, 2016</span></p>
...
```

  - RDFa makes it possible to re-use the ***content*** attribute of HTML

```
...
<p>Date: <span property="http://purl.org/dc/terms/created" content="2016-11-28">28th
of November, 2016</span></p>
...
```

  - also, ***content*** attribute can be useful when we define some statements through *meta* element (that may have no text content) in the header of the document

```
<head prefix="og: http://ogp.me/ns#" >
   ...
   <meta property="og:title" content="The Trouble with Bob" />
   ...
</head>
```

More of RDFa

- *RDFa Core* introduces attribute ***about*** that can be used as an alternative to ***resource*** in setting the context (the *subject* of the statement)
  - Attribute *resource* may be used to present subject or object of a statement

```
<div resource="/alice/posts/trouble">
   <h2 property="title">The trouble ...</h2>
   <h3 property="creator" resource="#me">Alice</h3>
</div>
```

Example: We need to set up a separate index page for all different blogs

```
<ul resource="/alice/posts/trouble">
  <li resource="/alice/posts/trouble" property="title">The trouble ...</li>
  <li resource="/alice/posts/jos" property="title">Jo's Barbecue</li>
</ul>
```

The combination of *property* and *resource* inside the same element would be considered as *predicate* and *object* and would generate a different statement than originally intended.

```
<ul>
  <li resource="/alice/posts/trouble"><span property="title">The trouble ...</span></li>
  <li resource="/alice/posts/jos"><span property="title">Jo's Barbecue</span></li>
</ul>
```

… Current solution becomes a little bit complicated. Therefore ***about*** could be used…

```
<ul>
  <li about="/alice/posts/trouble" property="title">The trouble ...</li>
  <li about="/alice/posts/jos" property="title">Jo's Barbecue</li>
  ...
</ul>
```

More of RDFa

- *RDFa Core* allows definition of datatypes and language tag

```
<span property="dc:date" datatype="xsd:gYear">2011</span>
<span property="dc:name" xml:lang="en">John</span>
```

- *RDFa Core* attribute ***rel*** can be used as an alternative to ***property***
  - In contrast to *property*, *rel never* considers the textual content of an element (or the value of the *content* attribute). Instead, if no clear target has been specified for a link via, e.g., a *resource* or an *href* attribute, the processor is supposed to go “down” and find one or more targets in the hierarchy and use those.


```
<div vocab="http://xmlns.com/foaf/0.1/" resource="#me">
   <ul>
      <li property="knows" resource="http://example.com/bob/#me" typeof="Person">
         ...
      </li>
      <li property="knows" resource="http://example.com/eve/#me" typeof="Person">
         ...
      </li>
   </ul>
</div>
```

```
<div vocab="http://xmlns.com/foaf/0.1/" resource="#me">
   <ul rel="knows">
      <li resource="http://example.com/bob/#me" typeof="Person">
         ...
      </li>
      <li resource="http://example.com/eve/#me" typeof="Person">
         ...
      </li>
   </ul>
</div>
```

More of RDFa


```
<html xmlns="http://www.w3.org/1999/xhtml"
      prefix="dbp: http://dbpedia.org/property/"
      prefix="dbp-owl: http://dbpedia.org/ontology/"
      prefix="dbr: http://dbpedia.org/resource/"
      prefix="foaf: http://xmlns.com/foaf/0.1/"
      prefix="xsd: http://www.w3.org/2001/XMLSchema#">
  <head>
    <title>Albert Einstein</title>
  </head>
  <body>
    <div about="dbr:Albert_Einstein">
      <span property="foaf:name">Albert Einstein</span>
      <span property="dbp:dateOfBirth" datatype="xsd:date">1879-03-14</span>
      <div rel="dbp:birthPlace" resource="dbp:German_Empire">
        <span property="dbp:conventionalLongName">the German Empire</span>
        <span rel="dbp-owl:capital" resource="dbr:Berlin" />
      </div>
    </div>
  </body>
</html>
```


```
@prefix …

dbr:Albert_Einstein foaf:name "Albert Einstein" .
dbr:Albert_Einstein dbp:dateOfBirth "1879-03-14"^^xsd:date .
dbr:Albert_Einstein dbp:birthPlace dbr:German_Empire .

dbr:German_Empire dbp:conventionalLongName "the German Empire" .
dbr:German_Empire dbp-owl:capital dbr:Berlin .
```

JSON-LD

- ***JSON*** (JavaScript Object Notation) is a lightweight data-interchange format that is completely language independent but uses conventions that are familiar to programmers of most of the programming languages (an *object* is an unordered set of name/value pairs).
- ***JSON-LD*** is a lightweight Linked Data format that extends JSON:
  - easy for humans to read and write
  - it is based on the already successful JSON format
  - provides a way to help JSON data interoperate at Web-scale


```
{
  "@context": "http://json-ld.org/contexts/person.jsonld",
  "@id": "http://dbpedia.org/resource/John_Lennon",
  "name": "John Lennon",
  "born": "1940-10-09",
  "spouse": "http://dbpedia.org/resource/Cynthia_Lennon"
}
```

- JSON for Linked Data: *http://json-ld.org*
- JSON-LD Playground: *http://json-ld.org/playground/index.html*
- JSON-LD parser/serializer for RDFLib (Python lib): *https://github.com/RDFLib/rdflib-jsonld*
- JSON-LD Processor and API implementation in JavaScript: *https://npmjs.org/package/jsonld*

JSON-LD

- Ambiguity

```
{   "name":"Oleksiy",
    "homepage":"http://users.jyu.fi/~olkhriye" }
```

```
{   "name":"olkhriye",
    "homepage":"http://users.jyu.fi/~olkhriye" }
```

- To be specific

```
{   "http://ex1.com/name":"Oleksiy",
    "http://ex1.com/homepage":"http://users.jyu.fi/~olkhriye" }
```

- To be very concise use ***JSON-LD Context***
  - used to define the short-hand names

```
{   "@Context":"http://ex1.com/myApp.jsonld",
    "name":"Oleksiy",
    "homepage":"http://users.jyu.fi/~olkhriye" }
```

```
{   "@Context":{ "schema":"http://schema.org/",
                 "name":"schema:givenName",
                 "homepage":"schema:url" },
    "name":"Oleksiy",
    "homepage":"http://users.jyu.fi/~olkhriye" }
```

JSON-LD

- ***JSON-LD Identifiers***
  - uniquely identifies *things*


```
{   "@Context" {
       "name":"http://schema.org/givenName",
       "homepage":"http://schema.org/url"
    },
    "@id":"http://people.com/OleksiyKhriyenko",
    "name":"Oleksiy",
    "homepage":{"@id":"http://users.jyu.fi/~olkhriye"}
}
```


```
{   "@Context" {
       "name":"http://schema.org/givenName",
       "homepage":{
          "@id":"http://schema.org/url",
          "@type":"@id" }
    },
    "@id":"http://people.com/OleksiyKhriyenko",
    "name":"Oleksiy",
    "homepage":"http://users.jyu.fi/~olkhriye"
}
```

JSON-LD

- ***JSON-LD Type***
  - sets the data type of a *node* or *typed value*


```
{   "@Context" {
       "name":"http://schema.org/givenName",
       "homepage":{"@id":"http://schema.org/url","@type":"@id"}
    },
    "@id":"http://people.com/OleksiyKhriyenko",
    "@type":"http://schema.org/Person",
    "name":"Oleksiy",
    "homepage":"http://users.jyu.fi/~olkhriye"
}
```

- ***JSON-LD Value***
  - specifies the data that is associated with a particular *property* in the graph


```
{   "@Context":"http://context-jsonld.com/person",
    "@id":"http://people.com/OleksiyKhriyenko",
    "@type":"http://schema.org/Person",
    "name":[
            {"@value":"Oleksiy"},
            …
          ] ,
    "birthDate":{"@value":"1981-08-13", "@type":"xsd:data"}
}
```

JSON-LD

- ***JSON-LD language***
  - specifies the language for a particular string value or the default language of a JSON-LD document


```
{  "@Context" {
      "schema":"http://schema.org/",
      "name_ua":{"@id":"schema:givenName", "@language":"ua"},
      "name_ru":{"@id":"schema:givenName", "@language":"ru"},
      "birthDate":{"@id":"schema:birthData", "@type":"xsd:date"}
   },
   "@id":"http://people.com/OleksiyKhriyenko",
   "@type":"http://schema.org/Person",
   "name_ua":"Oleksiy",
   "name_ru":"Aleksey",
   "birthDate":"1981-08-13"
}
```

JSON-LD

- ***JSON-LD arrays***
  - ***@container*** is used to set the default container type for a *term*
  - ***@list*** represents *ordered* collection of values
  - ***@set*** describes *unordered* set of values


```
{  "@context":{ ...
                "nick":{ "@id": "http://xmlns.com/foaf/0.1/nick",
                         "@container": "@list“ },
                "name":{ "@id": "http://xmlns.com/foaf/0.1/name",
                         "@container": "@set“ }
              },
   "@id": "http://example.org/people#joebob",
   "nick": [ "joe", "bob", "jaybee" ],
   "name": [ { “@value”:“John”,
               “@language”:“en” },
             { “@value”:“Jonie”,
               “@language”:“fr” }
           ],
   “homepage”: [ “http://users.jyu.fi/~joe”,
                 “http://examplepage.com/~bob”,
                 “http://myPage.org/~jaybee”
               ],
   ...
}
```

JSON-LD

- ***JSON-LD reverse property*** *allows bidirection in directed graph*


```
[ { "@id": "#john",
    "http://myontology.com/onto#name": “John“ },
  { "@id": "#peter",
    "http://myontology.com/onto#name": “Peter",
    "http://myontology.com/onto#parent": { "@id": "#John" } },
  { "@id": "#mary",
    "http://myontology.com/onto#name": “Mary",
    "http://myontology.com/onto#parent": { "@id": "#John" } }
]
```

```
{ "@id": "#john",
  "http://myontology.com/onto#name": “John",
  "@reverse": { "http://myontology.com/onto#parent": [
                  { "@id": "#peter",
                    "http://myontology.com/onto#name": “Peter" },
                  { "@id": "#mary",
                    "http://myontology.com/onto#name": “Mary" } ] }
}
```

```
{ "@context": { "name": "http://myontology.com/onto#name",
                "children": { "@reverse": "http://myontology.com/onto#parent" } },
  "@id": "#john",
  "name": “John",
  "children": [ { "@id": "#peter", "name": “Peter" },
                { "@id": "#mary", "name": “Mary" } ]
}
```

JSON-LD

- ***@base*** sets the base IRI against which *relative IRIs* are resolved

http://myJSON.com/document.jsonld

```
{ "@context": {
                    "label": "http://www.w3.org/2000/01/rdf-schema#label"
                },
  "@id": "",
  "label": "A simple document"
}
```

```
{ "@context": {   "@base": "http://myJSON.com/document.jsonld"   },
  "@id": "",
  "label": "A simple document"
}
```

- ***@vocab*** expands *properties* and *values* in `@type` with a common prefix IRI.

If certain keys *should not be expanded* using the vocabulary IRI, a term can be explicitly set to *null* in the context.

```
{ "@context": {
                    "@vocab": "http://schema.org/",
                    "dbID": null
                },
  "@id": "http://example.org/places#SalsaOrchidea",
  "@type": "Restaurant",
  "name": "Salsa Orchidea",
  "dbID": "12345678"
}
```

JSON-LD

- ***JSON-LD graph*** - used to group a set of *nodes*. make statements about a *graph* itself, rather than just a single *node*.


```
{ "@context": { "generatedAt": {
                    "@id": "http://www.w3.org/ns/prov#generatedAtTime",
                    "@type": "http://www.w3.org/2001/XMLSchema#date"
                  }
                },
  "@id": "http://example.org/graphs/13",
  "generatedAt": "2013-11-26",
  "@graph": [ { "@id": "http://example.org/about#manu",
                ... },
              { "@id": "http://http://example.org/foaf#me",
                ... }
            ]
}
```

- Explicit expression of default graph


```
{ "@context": ...,
  "@graph": [ {... },
              {... } ]
}
```

```
[ { "@context": ...,
    ... },
  { "@context": ...,
    ... } ]
```

JSON-LD

- ***JSON-LD compact IRI*** - expressing an IRI using a *prefix* and *suffix* separated by a colon (:):
- *prefix* matches a term defined within the *active context;*
- *suffix* does not begin with two slashes (//);
- if the *prefix* is not defined in the active context, or the *suffix* begins with two slashes (e.g. *http://example.com*), the value is interpreted as *absolute IRI* instead;
- if the *prefix* is an underscore (_), the value is interpreted as *blank node* identifier instead.


```
{
  "@context":
  {
    "foaf": "http://xmlns.com/foaf/0.1/"
  ...
  },
  "@type": "foaf:Person",
  "foaf:name": "Dave Longley",
  ...
}
```

JSON-LD and RDF

- ***JSON-LD***


```
{   "@Context":{
                    "schema":"http://schema.org/"
                },
    "@id":"http://people.com/OleksiyKhriyenko",
    "@type":"schema:Person",
    "schema:name":"Oleksiy",
    "schema:knows":{
          "@id":"http://people.com/JohnDou",
          "@type":"schema:Person",
          "schema:name":"John",
          "schema:knows":"http://people.com/OleksiyKhriyenko"
     }
}
```

- ***RDF***


```
@prefix schema: <http://schema.org/>.
<http://people.com/OleksiyKhriyenko> a schema:Person ;
          schema:name "Oleksiy" ;
          schema:knows <http://people.com/JohnDou> .
<http://people.com/JohnDou> a schema:Person ;
          schema:name "John" ;
          schema:knows <http://people.com/OleksiyKhriyenko> .
```

JSON-LD and HTML

- ***JSON-LD*** content can be easily embedded in HTML by placing it in a script element with the *type* attribute set to *application/ld+json*.


```
<script type="application/ld+json">
{
    "@Context":"http://context-jsonld.com/person",
    "@id":"http://people.com/OleksiyKhriyenko",
    "@type":"Person",
    "name":"Oleksiy",
    "knows":{
              "@id":"http://people.com/JohnDou",
              "@type":"Person",
              "name":"John",
              "knows":"http://people.com/OleksiyKhriyenko"
            }
}
</script>
```

JSON-LD and RDFa


```
<div prefix="foaf: http://xmlns.com/foaf/0.1/">
   <ul>
      <li typeof="foaf:Person">
        <a rel="foaf:homepage" href="http://example.com/bob/"
                                                property="foaf:name">Bob</a>
      </li>
      <li typeof="foaf:Person">
        <a rel="foaf:homepage" href="http://example.com/eve/"
                                                property="foaf:name">Eve</a>
      </li>
      <li typeof="foaf:Person">
        <a rel="foaf:homepage" href="http://example.com/manu/"
                                                property="foaf:name">Manu</a>
      </li>
   </ul>
</div>
```


```
{ "@context":{ "foaf": "http://xmlns.com/foaf/0.1/" },
  "@graph":[ { "@type": "foaf:Person",
               "foaf:homepage": "http://example.com/bob/",
               "foaf:name": "Bob" },
             { "@type": "foaf:Person",
               "foaf:homepage": "http://example.com/eve/",
               "foaf:name": "Eve" },
             { "@type": "foaf:Person",
               "foaf:homepage": "http://example.com/manu/",
               "foaf:name": "Manu"}
           ]
}
```

Part 2

Semantic Annotation

Semantic Annotation

Semantic Annotation enriches content with machine-processable information by linking background information to extracted concepts. These concepts, found in a document or another piece of content, are unambiguously defined and related to each other within and outside the content. (*http://ontotext.com/knowledgehub/fundamentals/semantic-annotation/*)

A typical process of *semantic enrichment* includes:

- *Text Identification* Text could be extracted from any form of unstructured data: articles, documents, non-textual sources such as PDF files, videos, voice recordings etc.
- *Text Analysis* Algorithms split sentences and identify concepts, such as people, things, places, events, numbers.
- *Concept Extraction* All recognized concepts are classified (they are defined as people, organizations, numbers etc.) and disambiguated (they are unambiguously defined according to a domain-specific knowledge base). For example, Rome is classified as a city and further disambiguated as Rome, Italy not Rome, Iowa. This is the most important stage of semantic annotation. It includes Named Entity Recognition and makes them machine-processable and understandable data pieces by linking them to a broader sets of already existing data.
- *Relationship Extraction* The relationships between the extracted concepts are identified and interlinked with related external or internal domain knowledge.
- *Indexing and storing in a semantic graph database* All the recognized and enriched with machine-readable data mentions of people, things, numbers etc. and the relationships between them are indexed and stored in a semantic graph database for further reference and use.

Related materials: *http://www.slideshare.net/dianamaynard/text-analysis-in-gate*

https://libraryconnect.elsevier.com/articles/knowledge-discovery-through-text-analytics-advances-challenges-and-opportunities

DBpedia Spotlight

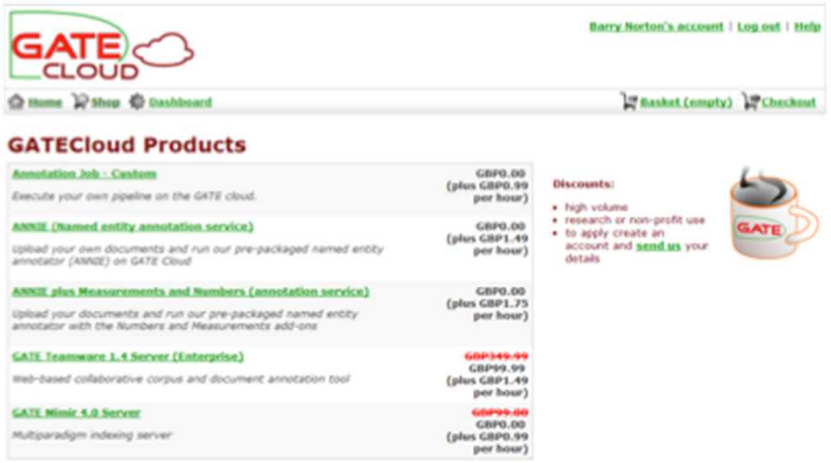
- ***DBpedia Spotlight*** is a tool for automatically annotating mentions of *DBpedia* resources in text, providing a solution for linking unstructured information sources to the Linked Open Data cloud through *DBpedia*. (*https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki*)
- Try out DBpedia Spotlight through Web Application or Web Service endpoints:
  - The *Web Application* is a user interface that allows you to enter text in a form and generates an HTML annotated version of the text with links to DBpedia.
  - The *Web Service endpoints* provide programmatic access to the demo, allowing you to retrieve data also in XML or JSON.
- Demo: *http://dbpedia-spotlight.github.io/demo/*

GATE

- ***GATE*** *(General Architecture for Text Engineering)* is an open-source framework for text engineering. Started in 1996, *GATE* has a large developer community and can be more readily customized for text annotation in different domains and for different purposes. *GATE* is used worldwide to build bespoke solutions by organizations including the Press Association and National Archive. Information extraction is supported in many languages. (*https://gate.ac.uk*)

- There is a possibility to build a *GATE processing pipeline* specifically for your domain. For this we take an RDF dataset and use this to produce what is called a *GATE Gazetteer*, which is a list of entities in a domain and associated text labels used to refer to those entities. We can produce a gazetteer using the RDF data from chosen domain.

- A *GATE pipeline* can be run locally or uploaded to the *GATE cloud*. Once set up, text can be submitted and then annotated using the domain specific data. The annotated text can then be output in a format such as RDFa. (*https://cloud.gate.ac.uk/*).



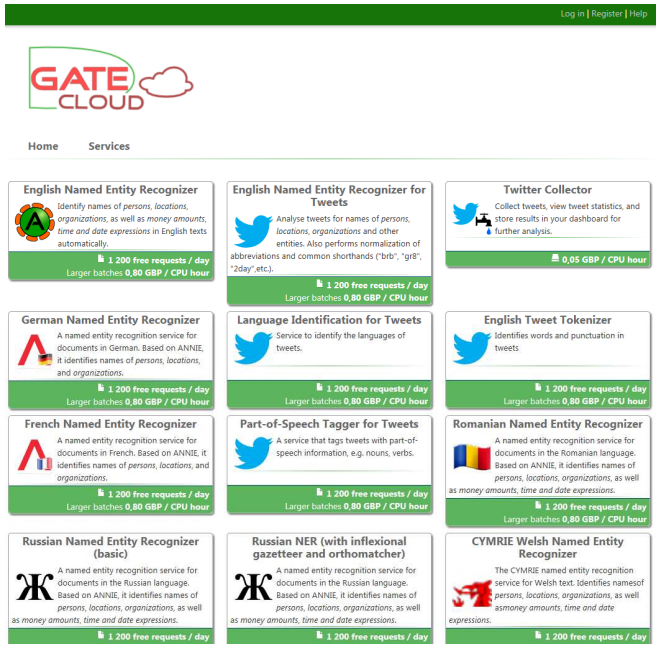
- *Related materials:*

http://www.slideshare.net/dianamaynard/text-analysis-in-gate

GATE Cloud

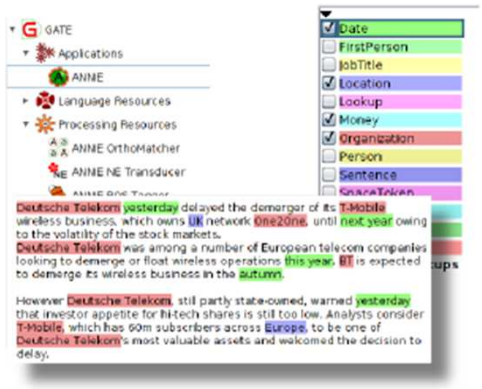
Text Analytics-as-a-Service. ***GATE Cloud*** – the home of affordable text analytics solutions from the world-leading open source ***GATE platform***. Collect and/or process documents and social media, using freemium pre-packaged annotation services, or scale out and run your own GATE pipeline on millions of documents. View and export the results in structured formats, or run your own private instance of our highly scalable GATE Mímir semantic search platform.

(*https://cloud.gate.ac.uk/*)

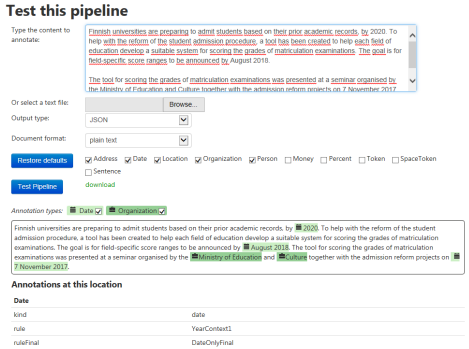


- Free Semantic Text Analysis APIs
- On-demand, Large-Scale Text Analytics
- Customized services and local cloud deployments

Set of ***services*** on GATE Cloud: *https://cloud.gate.ac.uk/shopfront*

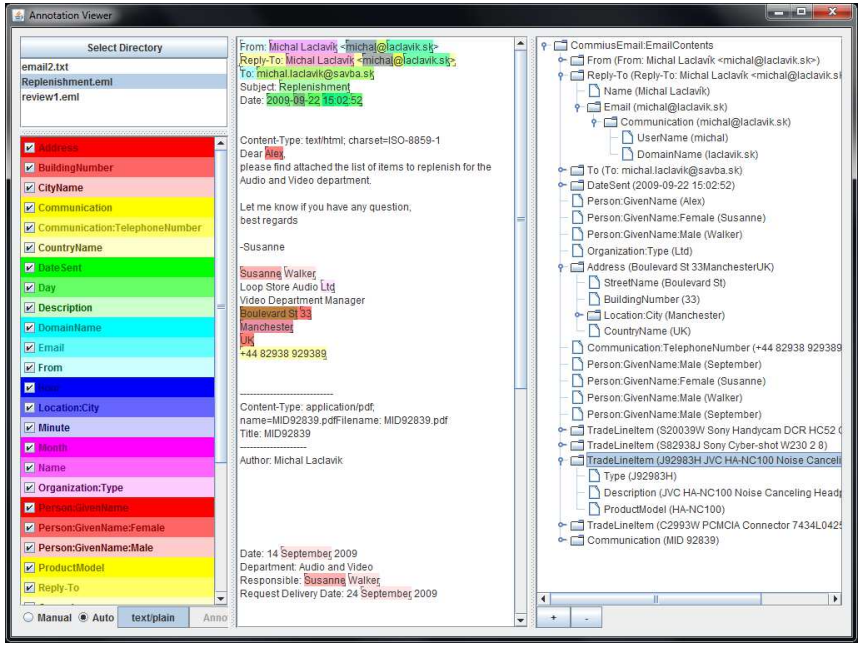


demo



OnTeA

- ***OnTeA*** *(Ontology based Text Annotation)* is a Pattern based Semantic Annotation Platform. OnTeA search or create semantic meta data from text or documents using pattern based approaches. The Platform contains also graphical user interface, which shows identified objects in the text of email message or text file. The Platform also analyses HTML, PDF and Word email attachments.
- OnTeA uses two main techniques for information extraction:
  - patterns based on regular expressions
  - gazetteers: place names, locations, days of the weeks, etc. (now working with GATE or OntoText gazetteers).
- Link: *http://ontea.sourceforge.net/*



OnTeA

- Motivation: *to extract semantic meta data from texts or documents…*
- Pattern based approach:
  - Unstructured content is also contains some patterns to be recognized…
  - Patterns are used to extract various objects that can be transformed to ontology class individuals and their properties…

Text	Patterns – regular expressions	Key - value
Nokia, Oy	*Company*: ([A-Za-z0-9]+)[,]+(Inc|Ltd|Oy)	*Company*: Nokia
info@dna.fi	*Email:* [-_.a-z0-9]+@[-_.a-zA-Z0-9]+\.[a-z]{2,8}	*Email*: info@dna.fi
Dr. Oleksiy Khriyenko	*Person:* (Mr.|Mrs.|Dr.) ([A-Z][a-z]+ [A-Z][a-z]+)	*Person*: Oleksiy Khriyenko
…	…	…

Example:

Text - ***Helsinki is the capital of Finland. Finland is in Europe.***

Patterns:

Location: *(in\|by) + (the)? *([A-Z][a-z]+)*

Country: *(capital of) + *([A-Z][a-z]+)*

City: *([A-Z][a-z]+) + (is) + (the)? + (capital of)*

Continent: <Country> *+ (is in) + (the)? *([A-Z][a-z]+)*

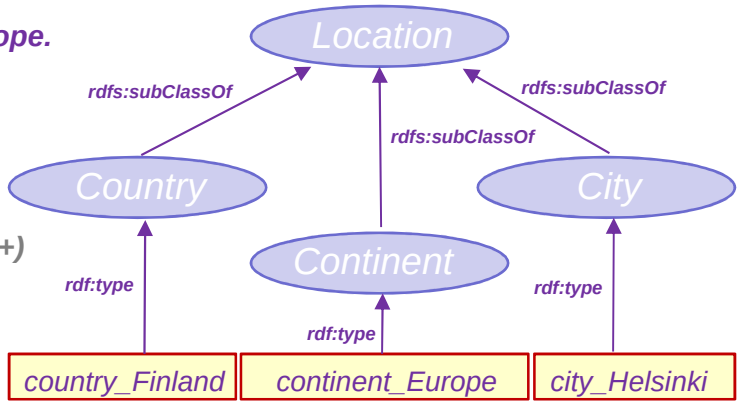
OnTeA key – value pairs:

Location: *Europe*

Country: *Finland*

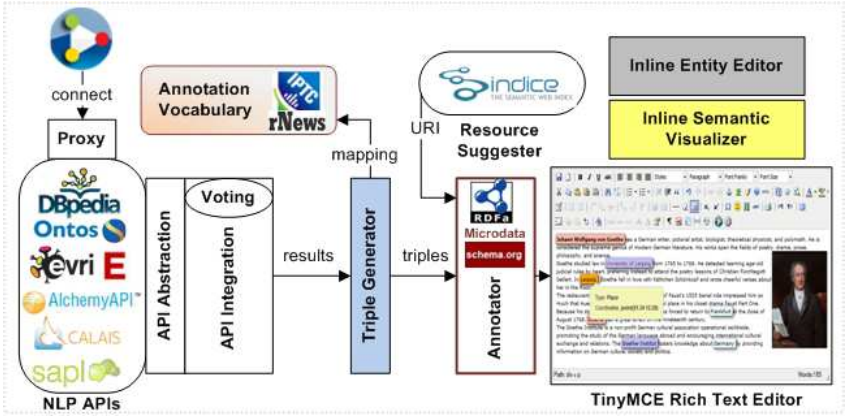
City: *Helsinki*

Continent: *Europe*



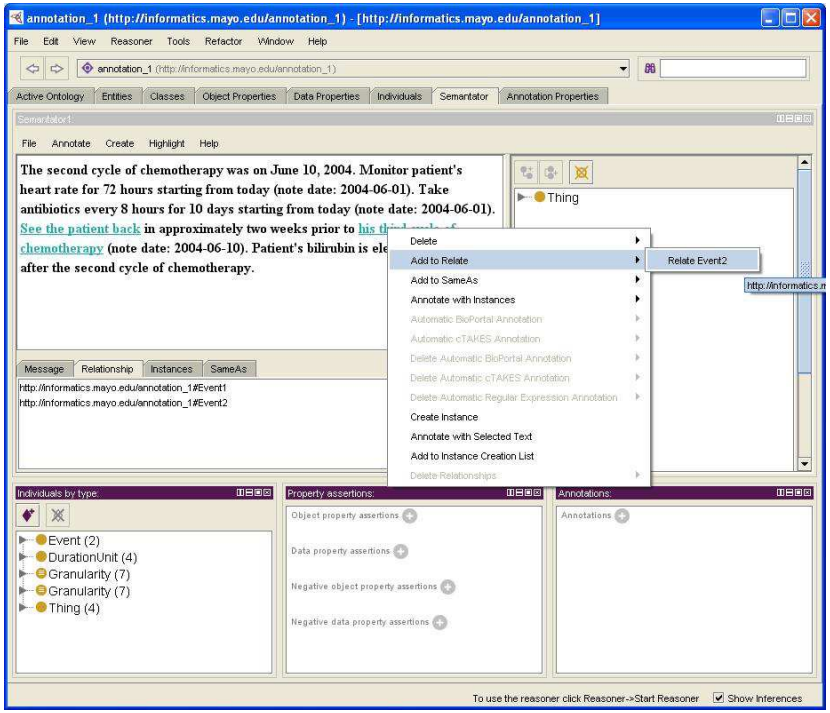
RDFaCE

- ***RDFaCE*** *(RDFa Content Editor)* is a Semantic content editor based on TinyMCE WYSIWYG editor. *RDFaCE* is an implementation for WYSIWYM (What You See Is What You Mean) concept (*http://youtu.be/wxtlAol4HB0*). WYSIWYM aims to enable end-users to easily annotate their content using *RDFa* and *Microdata* markups based on *Schema.org* vocabularies. (*http://ali1k.com/rdface/*) (*http://youtu.be/W5CdPq0C1GU*)(*http://aksw.org/Projects/RDFaCE.html*)
- *RDFaCE* supports automatic content annotation employing *Sindice*, *Swoogle* and *Prefix.cc* APIs for resource suggestion (providing appropriate URIs for subjects, properties and namespaces) as well as using external *NLP APIs* (*Alchemy, Extractiv, Open Calais, Ontos, Evri, Saplo, Lupedia* and *DBpedia spotlight*).
- Available as a plugin for *WordPress* blogging platform (*http://wordpress.org/plugins/rdface/*).
- *RDFaCE –lite* is based on lite-weight version of RDFa. It supports RDFa and Microdata on *rNews schema* (limited to news-specific metadata: person, location, organization entities, etc.)
- Demos: *http://ali1k.com/rdface/*



Semantator

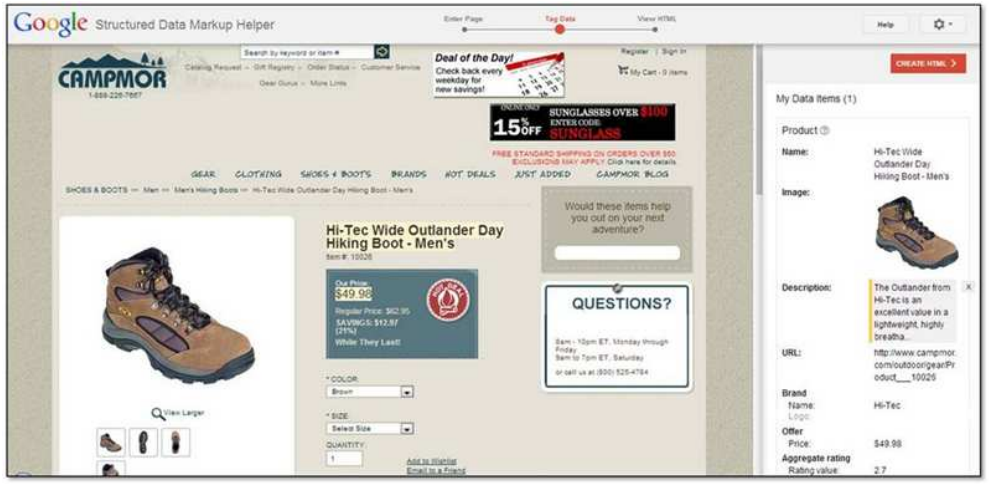
- ***Semantator*** *(Semantic Annotator)* is a tool developed in Mayo Clinic for users to semantically annotate data of interest with respect of domain ontologies in plain text. (*http://informatics.mayo.edu/CNTRO/index.php/Semantator*)
- *Semantator* is implemented as a Protege plug-in that allows users to view the ontology used for annotation, and the annotation results in the same environment. *Semantator* provides two modes:
  - *Manual annotation.* Expert can choose a document to be annotated and a domain ontology, highlight different pieces of information from the original text, and then mark which ontology concepts the information belongs to, link the instances together using the properties defined in the domain ontology.
  - *Semi-automatic annotation*. Users can choose to use different automatic annotation tools such as the National Center for Biomedical Ontologies (NCBO) annotator and Mayo Clinic's Clinical Text Analysis and Knowledge Extraction System (cTAKES), which are well-acknowledged tools for annotating biomedical and clinical text. Annotation results can be reviewed and modify as needed.



- Useful readings:
  - *http://www.sciencedirect.com/science/article/pii/S1532046413001020#*
  - *http://swat.cse.lehigh.edu/pubs/song12a.pdf*

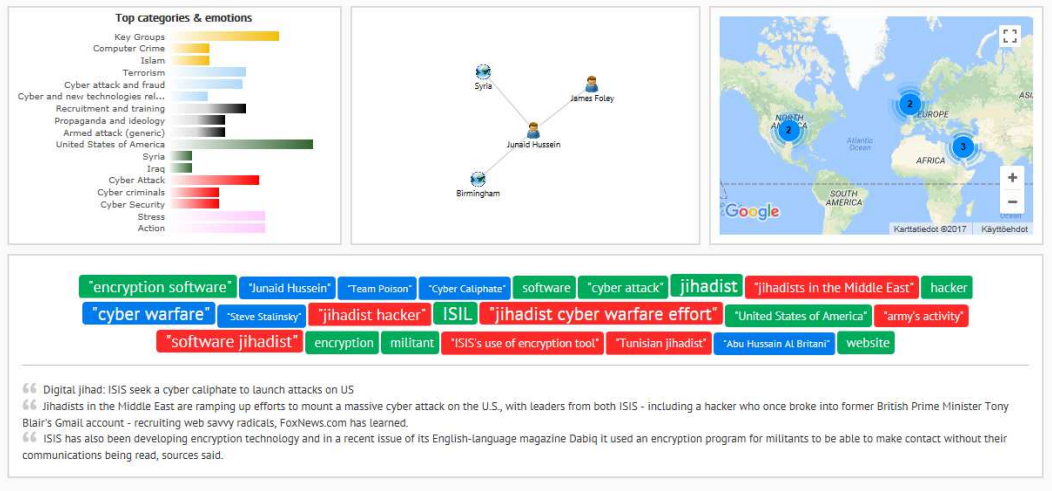
Structured Data Markup Helper

- ***Structured Data Markup Helper*** helps to update a site with on-page markup that enables search engines (e.g. Google, Bing, Yahoo!, Yandex) and other products to understand the information on web pages and provide richer search results in order to make it easier for users to find relevant information on the web. (*https://www.google.com/webmasters/markup-helper/*)
- Markup Helper uses *microdata* and *JSON-LD* formats with the *schema.org* vocabulary (a collaboration by Google, Microsoft, and Yahoo! to improve data description and interoperability on the web).
- Helpful links:
  - About Markup Helper: https://support.google.com/webmasters/answer/3069489?hl=en
  - Microdata: http://www.w3.org/TR/microdata/
  - JSON-LD: http://json-ld.org/
  - Schema.org: http://schema.org/



Cogito Intelligence API

- ***Cogito Intelligence API*** (*http://www.intelligenceapi.com/*) provides full semantic processing features—text mining (with time references (alpha version), semantic reasoning and inferential entities), categorization, semantic tagging, emotions, sentiment, fact mining, writeprint, and extraction relationships between entities that developers can easily integrate into their analysis platforms and applications for faster evaluation and analysis of documents, web pages, social media data or any big data sets or real-time information streams. The API comes in both SOAP XML-based and RESTful JSON-based flavors, and the features include:
  - **5 specific taxonomies** of terms (in over 1,000 different categories) for Intelligence, Terrorism, Cyber Crime, Crime and Geographic domains
  - A domain ontology **(updated regularly)** with a wide range of diverse topics, for example: weapons, crimes, cyber attacks, points of interest, chemical weapons, controlled substances, terrorist groups, critical infrastructure, world leaders, public companies and more



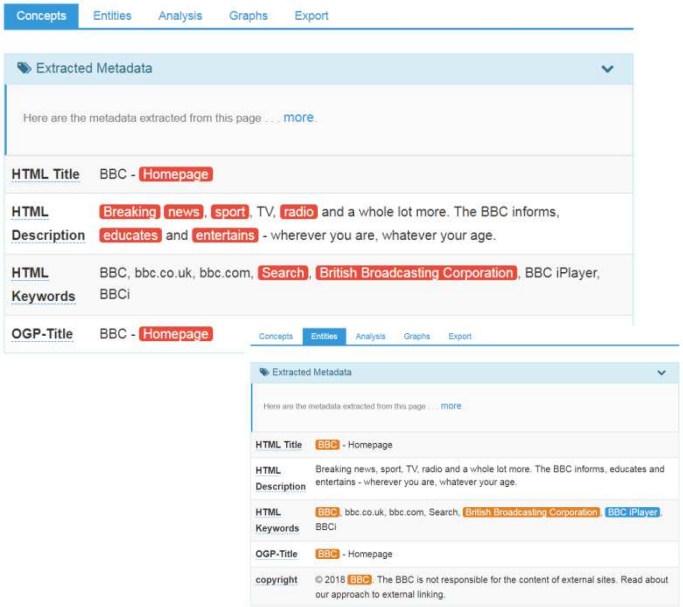
Live Demo: *http://www.intelligenceapi.com/demo/*

KBpedia

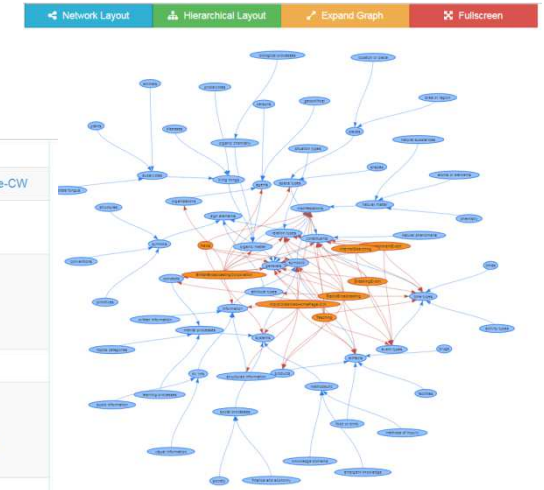
- ***KBpedia*** is a comprehensive knowledge structure for promoting data interoperability and knowledge-based artificial intelligence. It structure combines seven 'core' public knowledge bases: ***Wikipedia***, ***Wikidata***, ***schema.org***, ***DBpedia***, ***GeoNames***, ***OpenCyc***, and ***UMBEL*** — into an integrated whole. (*http://kbpedia.org/*)

  Live Demo: *http://kbpedia.org/demo/*

Exploits large-scale knowledge bases and semantic technologies for ***machine learning***, ***data interoperability and mapping***, and ***fact extraction and tagging***. It is a flexible and computable knowledge graph that can be sliced-and-diced and configured for all sorts of machine learning tasks, including *supervised*, *unsupervised* and *deep learning*.



Surface form	Score	Related [Ambiguous] Topics
homepage	0.77	http://kbpedia.org/kko/rc/WorldWideWebHomePage-CW
sport	0.58	http://kbpedia.org/kko/rc/Sports http://kbpedia.org/kko/rc/Sport
radio	0.58	http://kbpedia.org/kko/rc/RadioBroadcasting http://kbpedia.org/kko/rc/RadioCommunications http://kbpedia.org/kko/rc/RadioShow-IBT http://kbpedia.org/kko/rc/Radio
news	0.58	http://kbpedia.org/kko/rc/News
entertains	0.58	http://kbpedia.org/kko/rc/EntertainmentEvent http://kbpedia.org/kko/rc/Entertainment http://kbpedia.org/kko/rc/HostingASocialGathering http://kbpedia.org/kko/rc/Entertainers
educates	0.58	http://kbpedia.org/kko/rc/Teaching http://kbpedia.org/kko/rc/Education
breaking	0.58	http://kbpedia.org/kko/rc/BreakingEvent
search	0.50	http://kbpedia.org/kko/rc/InternetSearching http://kbpedia.org/kko/rc/LookingForSomething
british broadcasting corporation	0.50	http://kbpedia.org/kko/rc/BritishBroadcastingCorporation



Cognitive Services and APIs for NLP

- ***IBM Watson*** NLP related services on IBM Bluemix cloud (*https://www.ibm.com/watson/*):
  - Natural Language Understanding
  - Discovery
  - Tone Analyzer
  - Personality Insights
- ***Cogito API*** a Natural Language Processing API. It is a ready to deploy and fully configured API series that helps developers accelerate creation and deployment of unique applications that leverage large volumes of unstructured information from multiple sources. (*http://www.expertsystem.com/products/api-integrations/*).
- ***Dandelion API*** (Semantic Text Analytics as a service) From text to actionable data: extract meaning from unstructured text and put it in context with a simple API (including *entity* and *keywords/concepts extraction*, *content classification*, *sentiment analysis, semantic similarity, etc.*) (*https://dandelion.eu/*).
- ***Yahoo Content Analysis API*** lets users perform content analysis on text or a URL (*https://developer.yahoo.com/contentanalysis/, https://www.programmableweb.com/api/yahoo-content-analysis*).

Data Analytics Products

Products that address today's challenges of *complexity*, *heterogeneity* and *scalability* to enable *intelligent search*, *analysis* and *better decisions*.

Such products try to bring meaning to data:

- *analyze data extracted from heterogeneous sources*
- *create new knowledge via linking and fusion of data*
- *get meaning from data and answer the queries via intuitive data visualization*
- *explore data via interactive navigation*

Ontos EIGER comprises a suite of modules for integrating, linking, exploring and analyzing many different data silos. The back-end is based on the W3C standard, especially the linked data paradigm. The build in store serves as the enterprise knowledge graph or corporate knowledgebase. (*http://ontos.com/products/platform/*)

IBM Watson Analytics delivers cloud-based guided analytics, data visualization and predictive analytics that make understanding data easier. Watson Analytics offers a smart data discovery service available on the cloud, it guides data exploration, automates predictive analytics and enables effortless dashboard and infographic creation.

(*https://www.ibm.com/watson-analytics /*)

IBM Watson Explorer is an cognitive search and content analysis platform that gives access to insights from all the data that can be used to drive business performance and growth; search and analyze structured, unstructured, internal, external and public content to uncover trends and patterns that improve decision-making, customer service and return-on-investment; Leverage built-in machine learning, natural language processing and next-gen APIs to unlock hidden value in ALL data.

(*https://www.ibm.com/us-en/marketplace/content-analytics /*)

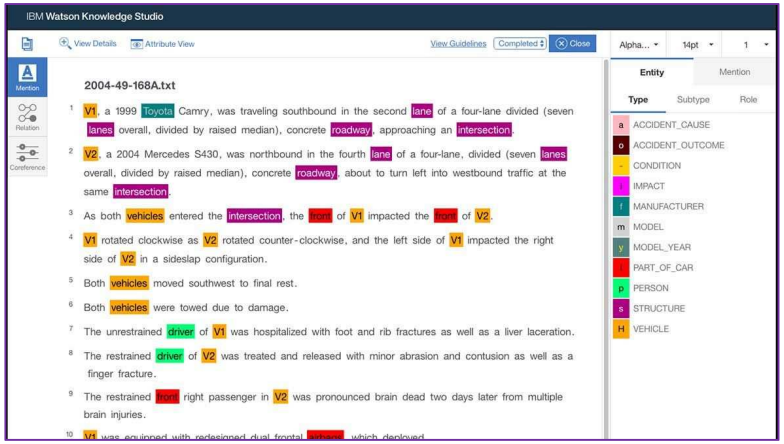
etc.

Watson Knowledge Studio

Watson can be taught to extract meaningful information from unstructured text. User can create annotators that later will be used by Watson to discover relationships in unstructured data.

Watson Knowledge Studio is a cloud-based application that enables developers and domain experts to collaborate and create custom annotator components for unique industries.

- *These annotators can identify mentions and relationships in unstructured data and be easily administered throughout their lifecycle using one common tool.*
- *Annotator components can be deployed directly to IBM Watson Explorer and Alchemy Language on IBM Watson Developer Cloud.*



Link: *https://www.ibm.com/marketplace/cloud/supervised-machine-learning/us/en-us*

https://www.ibm.com/blogs/watson/2016/06/alchemy-knowledge-studio/

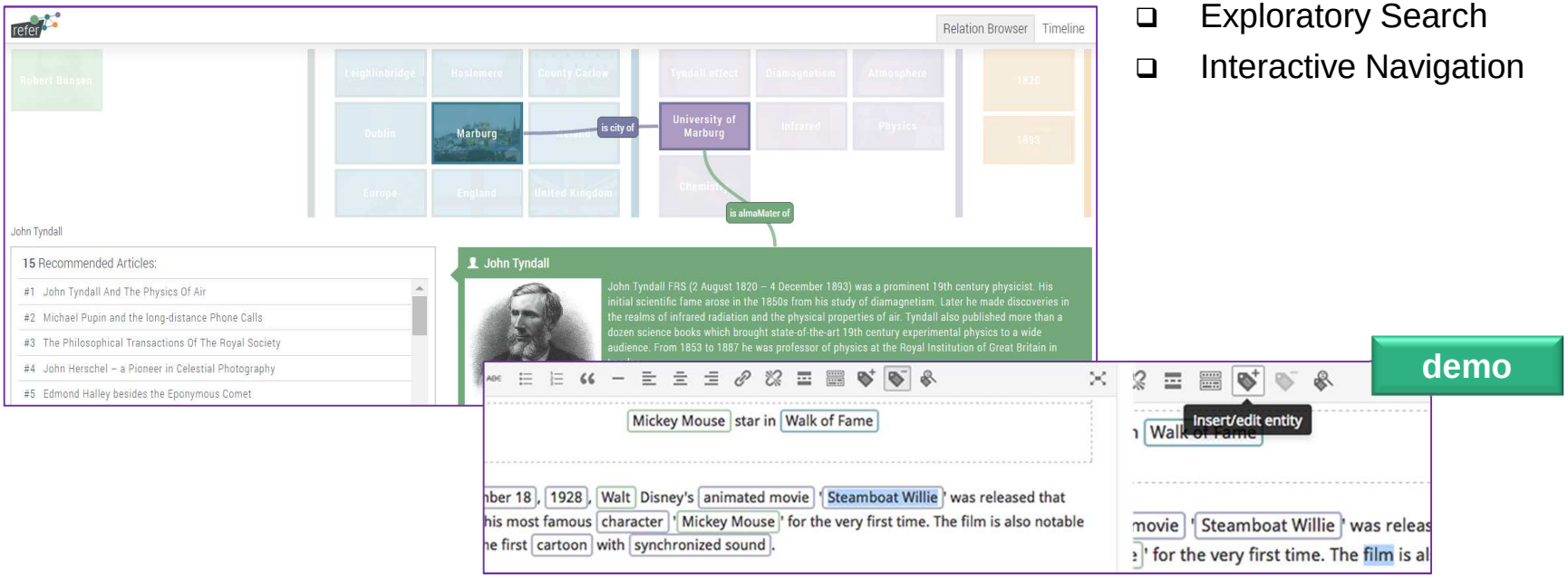
https://www.ibm.com/watson/developercloud/doc/wks/wks_overview.shtml

https://www.youtube.com/watch?v=xBoem605XQ4

Refer

Refer is an online-recommendation system based on Linked Open Data and Semantic Web Technologies. It aims to improve the user's and author's experience while curating and navigating in blogs, multimedia platforms, and archives through *Automated Annotation*, *Semantic Analysis*, *Content Enrichment* and *Relation Browser*. (*http://refer.cx/*)

- Content Analysis
- Exploratory Search
- Interactive Navigation

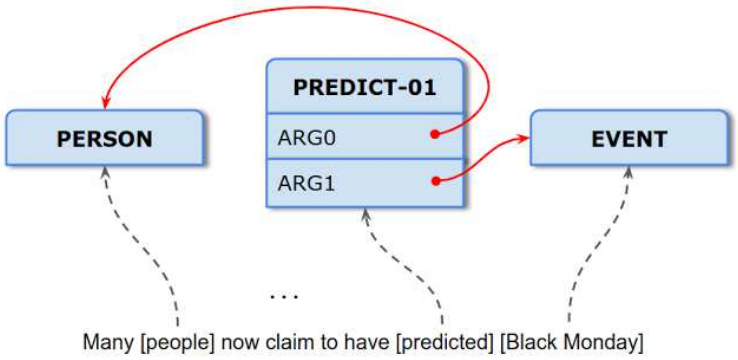


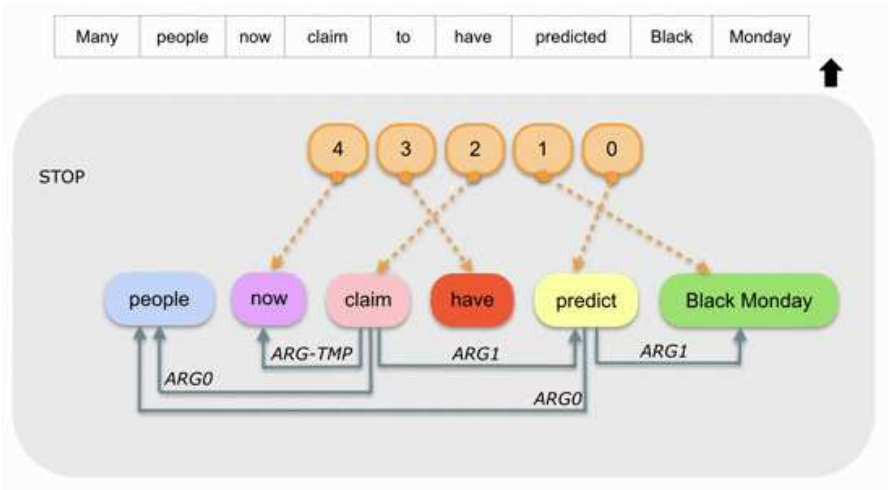
SLING

SLING is a Natural Language Frame Semantic Parser. Unlike the most practical natural language understanding (NLU) systems that used a pipeline of analysis stages, from part-of-speech tagging and dependency parsing to steps that computed a semantic representation of the input text, to avoid errors in earlier stages that would have cascading effects in later stages and the final representation, *SLING* parses natural language text directly into a representation of its meaning as a semantic frame graph. SLING uses a special-purpose *recurrent neural network* model to compute the output representation of input text through incremental editing operations on the frame graph. SLING's parser is trained using only the input words, bypassing the need for producing any intermediate annotations (e.g. dependency parses).

(*https://research.googleblog.com/2017/11/sling-natural-language-frame-semantic.html*)

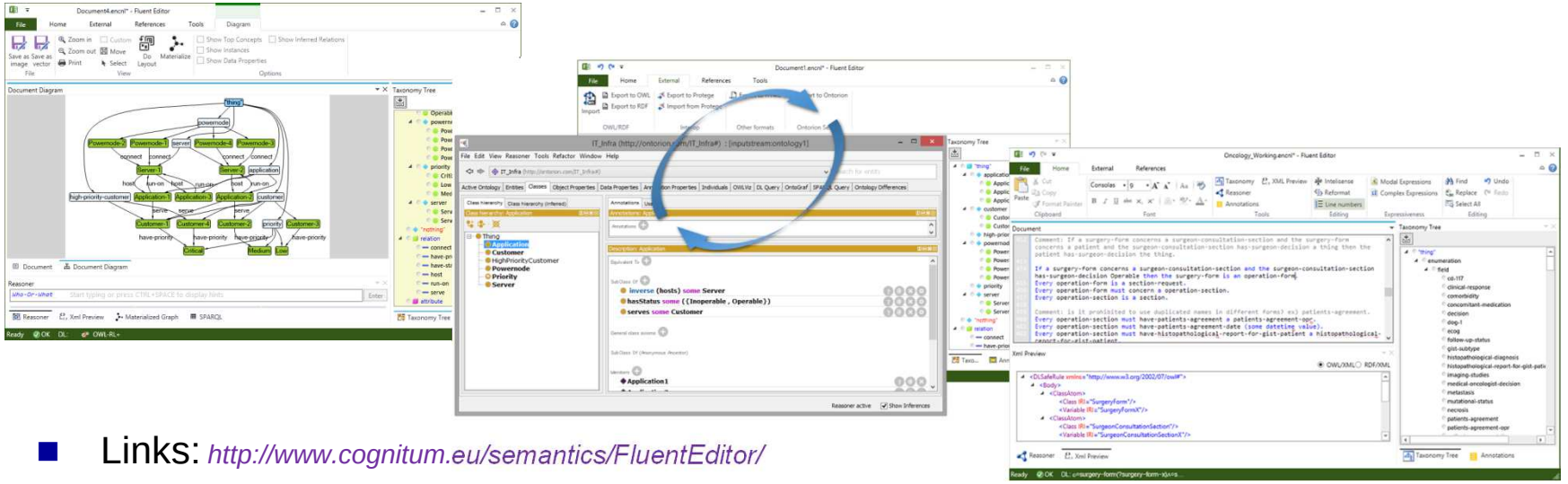
Source: *https://github.com/google/sling*





Fluent Editor

- ***Fluet Editor*** is an comprehensive tool for editing and manipulating complex ontologies that uses *Controlled Natural Language (CNL):*
  - allows natural-language driven ontology creation and editing;
  - provides a more user-friendly alternative to XML-based OWL editors;
  - uses of *Controlled English* as a *knowledge modeling language*;
  - supported via *Predictive Editor*, it stops the user from entering any sentence that is grammatically or morphologically incorrect and actively helps the user during sentence writing;
  - can be integrated with any other 3rd party tools compliant with W3C standards;
  - interoperable with *Protégé* and uses *R language package* to access an ontology from R environment.



- Links: *http://www.cognitum.eu/semantics/FluentEditor/*

Fluent Editor

- ***Controlled Natural Language***
  - Machine (web) processable;
  - Understandable to humans;
  - Restricted grammar and vocabulary;
  - Simplifies editing and structure;
  - Reduces ambiguity and errors;
  - Full OWL2, SWRL and SPARQL compatibility.
- ***Embedded reasoner***
  - Constantly checks knowledge base for consistency;
  - Generates XML automatically;
  - Corrects on the fly, preventing errors and speeding work pace;
  - Generates OWL files automatically.
- ***Predictive Editor***
  - Assists user with editing ontologies in real time;
  - Ensures perfect logic quality of ontologies;
  - Significantly increases editing speed;
  - Provides type-ahead hints and auto correct.
- ***With Fluent Editor™ business users can***
  - Import Existing ontologies from OWL files;
  - Export custom ontologies to OWL format;
  - Export custom rules to SWRL format;
  - Integrate custom software with CNL API;
  - Integrate with distributed semantic knowledge databases.

Fluent Editor

The same SWRL is verbalized in ***pure OWL/XML*** and ***Fluent Editor™***

```
<Body>
   <ClassAtom>
      <Class IRI="#Consent" />
      <Variable IRI="#Consent_0" />
   </ClassAtom>
   <ClassAtom>
      <Class IRI="#Patient" />
      <Variable IRI="#Patient_0" />
   </ClassAtom>
   <ClassAtom>
      <Class IRI="#Therapy" />
      <Variable IRI="#Therapy_0" />
   </ClassAtom>
   <ObjectPropertyAtom>
      <ObjectProperty IRI="#isRecommendedTo" />
      <Variable IRI="#Therapy_0" />
      <Variable IRI="#Patient_0" />
   </ObjectPropertyAtom>
   <ObjectPropertyAtom>
      <ObjectProperty IRI="#signs" />
      <Variable IRI="#Patient_0" />
      <Variable IRI="#Consent_0" />
   </ObjectPropertyAtom>
</Body>
<Head>
   <ObjectPropertyAtom>
      <ObjectProperty IRI="#isAppliedTo" />
      <Variable IRI="#Therapy_0" />
      <Variable IRI="#Patient_0" />
   </ObjectPropertyAtom>
</Head>
```

SWRL rule in Fluent Editor™:

```
If a patient signs a consent and a
therapy is-recommended-to the patient
then the therapy is-applied-to the
patient.
```

Asking questions in Fluent Editor™:

```
Who-Or-What is a city that belongs-to
Texas-State and has-latitude greater-
or-equal-to 0?
```

or

```
Who-Or-What is a customer that lives-
in a city that belongs-to California-
State and has-firstname equal-to
'John'?
```

Task 5