Module 9: Semantic Annotation and Ontologies
About this tutorial

- This tutorial will be a mixture of explanation, demos and hands-on work
- Things for you to try yourself are in red
- It assumes basic familiarity with the GATE GUI and with ANNIE and JAPE; no Java expertise
- Your hands-on materials are in module-9-hands-on.zip which extracts to module-9-hands-on/
  From: http://gate.ac.uk/wiki/TrainingCourseJune2013/
- Completing the hands-on tasks will help you in the exam....
We know the type of named entity but nothing more.

What kind of organization is Blackstone Group LP?

What is the job of William Hague?

Where is Eastern DRC, what does DRC stand for?

=> only semantics: choice of annotation type name

=> some knowledge hidden deep in JAPE & Code
Need More Semantics:

- To co-reference DRC with “Democratic Republic of Congo”
- To avoid scattered knowledge in JAPE/Java? Cities are locations, cities have zip codes, ...
- To disambiguate: which “Washington” (state / city)?
- To use extracted information to allow for queries like:
  - European politicians who visited an African country?
  - Politicians and actors travelling together?
- To use extracted information to add information to our own Database/Knowledgebase:
  - Add information about the buying-agreement to our data about Blackstone Group and First Potomac Realty Trust
  - Connect with trading information or other data we have
Semantic Annotation: Basic Idea/Vision

- Link annotations to concepts in a knowledge base.
- The annotated text is a “Mention” of a concept in the KB.
- We can use the knowledge associated with Mentions in our IE pipeline: e.g., Persons have JobTitles, Cities have zip codes.
- We can use the knowledge associated with Mentions for “Semantic Search”.
- We can use semantically annotated documents to add new facts to our knowledge base.

=> We need some way to represent knowledge.
Knowledge Base

Would want to represent knowledge for this domain:

- Westerwelle:
  has job Foreign minister of Germany → a politician
  Germany → a country, in Europe
  Member of the Free Democratic Party
  Free Democratic Party → a political party
  Political party → an organization

- Blackstone Group L.P. → a private equity company
  has NYSE symbol: BX
  based in: New York City
  New York City → a city
  located in: New York State which is located in USA
Use an ontology!
A formal way to represent knowledge as:

- Concepts of a domain or a set of domains
  “Agelina Jolie”, “Ghana”

- Relationships between concepts
  “New York City is located in New York State”

- Hierarchies of Concepts and Relationships
  “New York City is a City which is a Location”

- Associated Data
  “Blackstone Group has NYSE symbol BX”

- => most widely used formalism is RDF/OWL
OWL Ontologies - RDF(S)

- Based on RDF(S) - Resource Description Framework (Schema):
  - Everything is identified by an URI
  - Everything can be expressed as triples of the form
    **Subject Predicate Object:**
    :City rdfs:subClassOf :Location .
    :Location a rdfs:Class .
    :BlackstoneGroup :hasNyseSymbol “BX” .
  - Simple vocabulary to express things:
    rdf:type = “belongs to a class”
    rdf:Class = “the class of all classes”
    “BX” = the literal string “BX”
OWL Ontologies - URIs

- Nearly everything represented by URI (not blank nodes and literal values):
  http://my.ontology/locations#NewYorkCity

- URIs can look like URLs

- Often many URIs share the same prefix:
  http://my.ontology/locations#NewYorkState
  http://my.ontology/people#AngelinaJolie

- Common part http://my.ontology/ is “Base URI”, can abbreviate:
  locations#NewYorkState, people#AngelinaJolie

- Namespace + Fragment identifier
  loc: = http://my.ontology/locations#
  “:” alone can be used to indicate “default namespace”
  If default namespace is http://my.ontology/#
  → :Class1 really means http://my.ontology/#Class1
• All resources identified by URIs
  Different URIs may refer to the same resource

• Resources that are “Individuals” can be grouped into “Classes” and relate to other things and to values by “Properties”.

• Values represented through “Literals”:
  “BX” - a string (untyped literal)
  “New York State”@en – string with language tag (untyped)
  “Guido Westerwelle”^^xsd:string – typed literal
  “24”^^xsd:integer

• :A rdf:type :B – :A is contained in class :B
  :B rdf:type rdfs:Class – :B is an RDFS Class
  :B rdfs:subClassOf :C – all members of :B are in :C

=> Application may do class membership subsumption
OWL Ontologies

- Extend the vocabulary of RDF(S): more semantics, e.g.
  
  \texttt{owl:DatatypeProperty}
  \texttt{owl:Class} (different to rdfs:Class)
  \texttt{owl:sameAs}
  \texttt{owl:FunctionalProperty, owl:inverseOf}, ...

- (!) Reasoning/Inference: infer all derivable new facts from asserted facts

- Arbitrary RDFS/OWL is undecidable: restrict language!
  OWL Full = RDFS + semantics \rightarrow undecidable, incomplete
  OWL DL = decidable+complete but hard/slow
  OWL Lite = better than OWL DL (but still hard): GATE!

- OWL2: profiles EL, DL, QL, RL have better trade-off between expressiveness and performance
OWL Ontologies

• OWL: Web Ontology Language
• Classes/Concepts and Individuals/Instances
• Properties:
  DatatypeProperty: individual $\rightarrow$ literal
  ObjectProperty: individual $\rightarrow$ individual
  AnnotationProperty: resource $\rightarrow$ literal, but no inference
• Inference/Reasoning:
  - Inheritance/Subsumption (classes and properties)
  - “Restrictions”: domain, range, allValuesFrom, hasValue …infer class membership, property values (but: does not really “restrict” anything)
  Open World Assumption: what is not asserted, we do not now
  Non Unique Name Assumption: different names may be used for same entity
• Classes can have more than one parent, Individuals can belong to more than one class → OWL Ontologies are graphs, not trees
• Can be written down as RDF/XML, Turtle ...
OWL Ontologies – Inference Examples

• \( \text{:prop1 a owl:ObjectProperty .} \)
  \( \text{:prop1 rdfs:domain :Class1 .} \)
  \( \text{:A :prop1 :B .} \)
  \( \rightarrow \text{:A must be a member of :Class1} \)

• \( \text{:prop1 rdfs:range :Class2 .} \)
  \( \text{:A prop1 :D .} \)
  \( \rightarrow \text{:D must be a member of :Class2} \)

• \( \text{:prop2 a owl:FunctionalProperty .} \)
  \( \text{:A :prop2 :B .} \)
  \( \text{:A :prop2 :C .} \)
  \( \rightarrow \text{:B and :C must be the same!} \)

Different literal values: inconsistent (but not in GATE)
OWL Ontologies – Inference Examples

- \texttt{:prop3 a owl:TransitiveProperty .}
- \texttt{:A :prop3 :B .}
- \texttt{:B :prop3 :C .}
  \rightarrow \texttt{:A :prop3 :C .}

- \texttt{:prop4 a owl:ObjectProperty .}
- \texttt{:prop5 rdfs:subPropertyOf :prop4 .}
- \texttt{:A :prop5 :B .}
  \rightarrow \texttt{:A :prop4 :B .}

- \texttt{:prop6 owl:inverseOf :prop5 .}
  \rightarrow \texttt{:B :prop6 :A .}
OWL vs. Object Oriented

- Similar terminology but very different!
- Classes are not prototypes but merely sets of individuals defined by intension (rule) or extension (enumeration)
- Properties are “global” by default
- No real inheritance of property/value from classes to individuals
- No “real” restrictions/limitations but inference / inconsistency
- Inference often goes in a direction that is surprising or unexpected
OWL: Inference of values

- Cannot just add the property to the class and “inherit” in individual!
  (can add annotation property, but this will not be used to infer anything for the individuals of the class)

- Use a `owl:hasValue “Restriction”`:

  ```
  :Human a owl:Class ;
  rdfs:subClassOf [ 
    a owl:Restriction ;
    owl:onProperty :numberOfLegs ;
    owl:hasValue 2
  ] .
  :Person0213 a :Human .
  → :Person0213 :numberOfLegs 2 .
  ```
Ontologies in GATE

- Can use OWL-Lite ontologies as language resources (→ Plugin Ontology)
- Ontology Editor, Ontology Annotation Tool, Relation Annotation Tool (→ Plugin Ontology_Tools)
- Ontology-enabled JAPE, JAPE Plus
- LKB Gazetteer (→ Plugin Gazetteer_LKB) OntoRoot Gazetteer (→ Plugin Gazetteer_Ontology_Based)
- Ontology-based evaluation (→ Plugin Ontology_BDM_Computation)
- Java API for ontology manipulation, triple manipulation, SPARQL queries
- Simple CLI commands for ontology handling, querying, SPARQL
GATE Ontology Implementation

- Based on Sesame and the OWLIM-Lite SAIL (Storage and Inference Layer) implementation from Ontotext
- Fast in memory repository, scales to millions of statements (depending on RAM)
- In addition to local file ontology, can connect to server:
  - OWLIM Lite
  - OWLIM SE/Enterprise: commercial product, persistent and scalable implementation for huge (billion triples) ontologies
- Supports “almost OWL-Lite”
- Java API represents OWL concepts (ontology, property, literal) as Java objects
  Also provides support for SPARQL and manipulating Triples directly
Load Ontology

- Need plugin Ontology
- For Editor, also need plugin Ontology_Tools
- Language Resource → New → OWLIM Ontology

![Parameters for the new OWLIM Ontology](image)

- Loaded:

![Language Resources](image)
Ontology Viewer/Editor

- Basic viewing of ontologies
- Some edit functionalities:
  - create new concepts and instances
  - define new properties and property values
  - deletion
- Some limitations of what's supported, basically chosen from practical needs for semantic annotation
- Not a Protégé replacement
Ontology Editor
URIs, Labels, Comments

• The names of classes, properties or instances shown in the GUI are the fragment identifiers of their URIs
  http://gate.ac.uk/example#Person → “Person”

• URIs and fragment identifiers cannot contain spaces and certain other characters: use underscore or “encode” (%20)

• To also store the correctly spelled name (or several), the annotation property “label” is often used:
  • right click on the class-instance → Properties → label, enter the value in the dialogue box (cannot chose type or language!)

• The comment property is often used for documentation purposes, also a string

• Comments and labels are annotation properties: no inference but can be used with properties and classes too
New label

“lexicalisation”
Hands-on 1: classes and individuals

- Load the Ontology and Ontology_Tools plugins
- Language Resource → New → OWLIM Ontology
  - For RdfXmlURL use test-ontology.owl
  - This loads a small ontology of Entity, Location, etc.
- Double-click on the ontology LR to open the Viewer
- Create a subclass of “Location” called “City” and then add the city where you live as an instance of “City”
- Add yourself as an individual of the class “Person”
- Add a label with your full name
- Save the ontology (right click on ontology in resources pane and select “Save as”)
- Keep the ontology open for the next hands on
Datatype Properties

- Datatype properties link individuals to data values
- Datatype properties can (but do not have to) be of type boolean, date, int, ...
- Available datatypes taken from XMLSchema
- To define a new data property in the Ontology Editor
  - Select an ontology class and click on the D button
  - Choose the desired datatype from the list (e.g. xsd:int)
  - Provide the property name (e.g. hasAge)
  - Specify the domain (the class of the individuals having this property) (no domain: domain is owl:Thing)
  - If more than one class is listed as a domain, this asserts that any individual having that property must be a member of the intersection of those classes
Adding a new property

- Name Space: http://gate.ac.uk/example#
- Data Type: http://www.w3.org/2001/XMLSchema#positiveInteger
- Property Name: hasAge
Adding a DatatypeProperty Value

- To add a value for an instance, right click on the instance and select “Properties” and then the name of the property for which you want to add a value.
- If the property is not listed, then you haven't defined it yet for the concept to which your instance belongs.
- Enter the value in the popup box. GATE does some basic type checking (not OWL!)
- You should now see the property and its value listed in the right hand pane.
- The same property can be added multiple times with different values (but not the same value). GATE does **not** prevent you from doing this for functional properties → inconsistent ontology!
Adding a property value

1. Select instance and property
2. Add or select value
3. Property and value displayed
Hands-on 2: Datatype properties

- Use the ontology from the previous exercise
- Add a datatype property “hasAge” with domain “Person” and domain “xsd:nonNegativeInteger”
- Add a value for the hasAge property to the instance of Person that refers to you (you can make it up if you don't want to reveal your real age!)
- Add an instance of Organization denoting the organization you work for (make one up if you like)
- Save the ontology (with the same name as before)
- Keep everything open for the next hands-on
Object Properties

- Object properties describe relationships between individuals, e.g. people work for organisations.
- Domain is the subject of the relation (the thing it applies to).
- Range is the object of the relation (the possible "values").

Similar to domains, multiple classes for a range will assert that the value will belong to the intersection of all specified classes.
Creating new Object Properties

To define a new object property:

- Click on the O button
- Provide a property name and (optionally) domain / range

To set the value of an object property for an instance:

- Right-click on the instance
- Select Properties and then the name of the relevant property
- From the drop down list of instances, choose the correct instance as a value and add to the list of values

GATE does not prevent adding multiple values for a functional property: → individuals are same or ontology inconsistent!
New Object Property

[Diagram showing a tree structure with nodes labeled as Classes and Instances, Properties, Resource Information, Direct Super Classes, and All Super Classes. Each node is connected with arrows indicating hierarchy. There is a dialog box showing the process for defining a new object property, with fields for Name Space, Property Name, Domain, and Range.]
Hands-on 3: Object properties

- Use the entity ontology you saved in the previous exercise
- Add an object property “worksFor” to model that persons work for organizations: which domain / range?
- Add a property value so you work for the organization you added earlier
- Right click the ontology and choose “Load”
  As file select “employs.turtle”
  As file format choose “turtle”
  This loads a file that contains:
    :employs owl:inverseOf :worksFor .
- Click the organization instance and check that “employs” got automatically inferred from “worksFor”
- Save the ontology
Using Ontologies on a Server

- Use the ConnectSesameOntology LR
- Can connect to any remote Sesame repository, but GATE only fully supports OWLIM with ruleset owl-max
- Parameters repositoryID and repositoryLocation:
  - repositoryID: someid
  - repositoryLocation: http://uid:pwd@host.com:8080/openrdf-sesame/repositories/someid
- Do NOT use the editor/viewer or other GUI tools with this unless the ontology is small!
- Mainly for use with the Java API
- If not an OWLIM/owl-max repository, some things may still be possible with the Java API
Ontology Design Principles

• There are many ways to encode a domain in an ontology – use your application needs as a guide. Keep it simple: only model what is needed, not what is true.

• Ontology authoring is often iterative and evolves with your text analysis application

• Classes vs. instances: this can vary, but as a rough guide, proper nouns are usually instances, common nouns are usually classes. Dilemma: OWL-Lite cannot treat something as both a Class and an Instance

• Level of granularity: what subclasses do you need? (e.g do organisations need subclasses such as government, education, charity?)

• Domains and ranges: really only useful when the inference is needed! Similar for local range restrictions \(\textit{allValuesFrom}, \textit{someValuesFrom}\)

• Properties: subproperties, transitive properties, inverse properties can be useful

• Literals: make sure literals are always typed or never typed
Semantic Annotation

• Link text mentions to ontology resources: Mention annotations have a feature (inst) with the URI of the resource

• Usually linking to individuals, may link to classes

• Use:
  • Information Extraction (OBIE: Ontology-Based Information Extraction): e.g. match a Vegetable with a Plant in JAPE, add knowledge useful for IE
  • Semantic Search
  • Knowledge Acquisition:
    Ontology Population: add facts to given structure
    Ontology Learning: find structure of ontology too
Semantic Annotation

- Match lexical information (e.g. value of rdfs:label property) with text / word stems / lemmata
- Must disambiguate between possible alternatives
  “bank” (river) vs. “bank” (institution)
  “G. Bush” (father) vs. “G. Bush” (son) vs. “G. Bush” (not related)
  → Knowledge from the ontology may be useful here
- Link disambiguated mentions to ontology via URI
Semantic Annotation

Greece v Argentina: Who wins on penalties?
By Robert Plummer Business reporter, BBC News

Anyone examining the precedents for the Greek financial crisis might well be amused by the draw for next month’s football World Cup matches.

Greece’s players celebrated after qualifying for the 2010 World Cup.

For, as fate would have it, Greece’s foes in Group B include the country that last suffered a comparable economic fiasco: Argentina.

In the worst-case scenario, Argentina’s recent past is Greece’s future.

The peso collapse, massive default and subsequent social and political unrest that rocked Argentina in 2001-2002 are being seen by many economists as an awful warning for the politicians in Athens and Brussels.

As far as football is concerned, this and final group match.

But the day of decision for the Greeks is upon them.

The EU and the IMF have agreed to kick off with a loan to help

Print

Location

<table>
<thead>
<tr>
<th>Type</th>
<th>Set</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>1222</td>
<td>1228</td>
<td>1228</td>
</tr>
<tr>
<td>Location</td>
<td>1222</td>
<td>1228</td>
<td>1228</td>
</tr>
<tr>
<td>Location</td>
<td>1222</td>
<td>1228</td>
<td>1228</td>
</tr>
<tr>
<td>Location</td>
<td>1222</td>
<td>1228</td>
<td>1228</td>
</tr>
<tr>
<td>Location</td>
<td>1222</td>
<td>1228</td>
<td>1228</td>
</tr>
<tr>
<td>Location</td>
<td>1222</td>
<td>1228</td>
<td>1228</td>
</tr>
<tr>
<td>Location</td>
<td>1233</td>
<td>1241</td>
<td>1241</td>
</tr>
<tr>
<td>Organization</td>
<td>1556</td>
<td>1558</td>
<td>1558</td>
</tr>
</tbody>
</table>
• Ontology Population: add new facts to a given ontology. The ontology structure and many classes and individuals are already there:
  “Westerwelle visits Ghana”
  \[ \text{:GWesterwelle01} \text{ :actorOf :Event001} . \]
  \[ \text{:Event001} \text{ a :VisitingEvent} . \]
  \[ \text{:Event001} \text{ :destination :Ghana} . \]
  …

• Ontology Learning: also create or extend the structure of the ontology.
Semantic Annotation: How

- Manually
  GATE: ontology based annotation using OAT/RAT (Ontology Annotation Tool, Relation Annotation Tool)

- Automatically
  - Gazetteer/rule/pattern based
    GATE: OntoRoot gazetteer, LKB gazetteer, JAPE, ...
  - Classifier (ML) based
  - Combination of the two
Manual semantic annotation: OAT

- Shows document and ontology class hierarchy side-by-side
- Interactive creation of annotations that link to the ontology class/instance
- Allows on-the-fly instance creation
- Used to create evaluation or training corpus
- Plugin: Ontology_Tools
  Adds a button “OAT” in the document view
Customisation has to be done for each document.

To ensure that any new instances automatically have a label (the string you selected in the document), tick Select text as property value.

To put all annotations into a set other than Default, change accordingly.

By default, OAT creates:
- Annotations of type Mention
- Class feature with the class URI
- Inst feature with the instance URI
As well as picking MPs for Westminster, voters will elect councillors in 164 local authorities across England.

Voting in the general election will take place in 649 constituencies, with nearly 4,150 candidates standing for election across the country.

David Cameron was the first of the main UK party leaders to cast their vote. The Tory leader went to a community hall in Witney, Oxfordshire, shortly after 1030 BST, accompanied by his wife Samantha.

Labour leader Gordon Brown went to vote shortly after 1100 BST at a community centre close to his home in North Queensferry, Fife. His wife Sarah was with him.

Nick Clegg, leader of the Liberal Democrats, arrived at a polling station in Sheffield Hallam at 1120 BST. His wife Miriam is unable to vote in the general election because she is a Spanish citizen.

The leader of the Scottish National Party, Alex Salmond, cast his vote shortly before noon, at Macduff in Banffshire, leading in the constituency of Ynys Mon in north Wales.

Polling in one constituency - Thirsk and Malton - was delayed, due to the death of one of the candidates.

<table>
<thead>
<tr>
<th>Type</th>
<th>SetStart</th>
<th>End</th>
<th>Id</th>
<th>{class=<a href="http://www.elephant.org.uk/skills/ontologies/ontologies.owl%7D">http://www.elephant.org.uk/skills/ontologies/ontologies.owl}</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mention</td>
<td>1277</td>
<td>1289</td>
<td>55</td>
<td>{class=<a href="http://www.elephant.org.uk/skills/ontologies/ontologies.owl%7D">http://www.elephant.org.uk/skills/ontologies/ontologies.owl}</a></td>
</tr>
</tbody>
</table>
OAT: The Editor Pop-up

![Diagram showing a list of named entities with labels such as Person, Leader, Nick Clegg, Gordon Brown, David Cameron, Leader_0002A, Leader_0002B, Leader_0002C. Options include Apply To All, Create Instance, Dehighlight.](image)
Annotating classes and instances

- Be careful about the difference between annotating classes and instances
- If you want to add UK to the ontology as an instance of a Location, you need to select “Create instance”
- Note that this will create a new instance in the ontology with a name like Location_00020. The string UK will appear as a label on that instance.
- If you just want to annotate UK with the class Location, then deselect “Create instance”
Annotating a class

Annotating UK as a class will create a new label on the class with the text string → not what we want!

BUT: we may want to annotate “location” that way ...
Annotating an instance

Annotating UK as an instance will create a new instance of Location and set the label of the instance to the text string
Hands-on 4: using OAT

- Use the previously created ontology
- Load the document `voting-example.xml` (from hands-on)
- Select the OAT button from the doc viewer
- From the Options tab, set “Key” as the annotation set and tick “Select text as property value”
- Annotate every instance of UK in the text as an instance of a Location
- **Tip:** Make sure you select “Create instance” and “Apply to all” before choosing the target class
- Switch to the ontology viewer to see the new instance
- Examine the annotations created in Key and their features
- Save the ontology and the document
OAT: Comments

- The options to filter out some classes / only show some are useful when working with bigger ontologies
- Limitation: cannot annotate properties:
  - → RAT (Relation Annotation Tool)
Relation Annotation Tool (RAT)

- RAT annotates a document with ontology instances and creates relations between annotations by means of ontology object properties.
- It is compatible with OAT, but focuses on relations between annotations modelled as object properties.
- Plugin Ontology_Tools
- It is comprised of 2 viewers: RATC (RAT-Concept) and RATI (RAT-Instance).
- Buttons RATC and RATI in document editor work in tandem.
- The RATC pane (on the RHS) looks similar to OAT. Click the checkbox beside a class to display the relevant instances.
The RAT-I view (lower horizontal pane) shows two columns: one for instances and one for properties.

To create a new instance, select an item in the ontology and then select the relevant text in the document.

Click “New instance”.

Any properties on the relevant class will be shown on the RHS of the table.

To add a property range, select a property and choose a value from the dropdown list.

Only object properties will be shown: it is not possible to add datatype properties in this way.
Hands-on 5: RAT

- Use the document from the previous hands-on
- Load the ontology test-ontology-instances.owl and remove the old ontology
- Click on RAT-C or RAT-I to display the viewers
- Add a new instance Liberal Democrats to the class Organization
- Add a new instance Nick Clegg to the class Leader
- Select the Nick Clegg instance and add the value of the person_works_for property to Liberal Democrats
- Use the ontology viewer to check the results, then save the ontology (may need to select a different instance then Nick_Clegg to update view)
Adding a property value

- Your result should look something like this:
Checking the result

Check that the instance and property have been added correctly, by viewing it in the ontology editor
RAT Comments

• Tool to add individuals and object properties that model relationships between them based on document text

• Limitation: cannot model relations as individuals, but often we need to model n-ary relations, actions, events ..
OAT vs RAT

- In OAT, you have the option to annotate all mentions of the selected string in one go, e.g. the string “Liberal Democrats” as being the mention of the respective instance from the ontology. In RAT, you'll have to annotate each of the occurrences of this string over and over again.

- OAT currently creates rather opaque instance URIs (e.g., Leader_0007A with label “David Cameron”), so once you have several automatically created instances of the same class, it becomes hard to distinguish which is which in OAT. RAT shows you all labels, not just the URI, so it's easier to select.

- In OAT you can annotate a string as a mention of a class, without giving an instance.
GATE: Automatic Semantic Annotation

- Ontology aware Gazetteers:
  - OntoRoot gazetteer
  - LKB Gazetteer
  - Other gazetteers, using inst/class features
- Ontology aware JAPE
- Semantic Enrichment: LKB Gazetteer, JAPE
Ontology Lookup: OntoRoot Gazetteer

- Finds mentions in the text matching classes, instances, data property values and labels in the ontology
- Matching can be done between any morphological or typographical variant (e.g. upper/lower case, CamelCase)
- Converts CamelCase names, hyphens, underscores
- Morphological analysis is performed on both text and ontology, then matching is done between the two at the root level.
- Text is annotated with features containing the root and original string(s)
- Creates a gazetteer PR that can be used with the FlexibleGazetteerPR
OntoRoot Gazetteer

- Lives in the Gazetteer_Ontology_Based plugin
- Generates candidate gazetteer list from ontology
- Runs the Tokeniser, POS tagger, Morphological Analyser to create lemmas from the labels and the fragment identifiers of all classes and instances and then creates lists to match against the text
- Gordon_Brown, GordonBrown → Gordon Brown
- Note that the gazetteer produced is stored in memory only and cannot be edited → limited size!
- Must use tokeniser, sentence splitter, POS tagger and morphological analyser first: so we get “root” (lemma) feature!
### Init-time OntoRoot params

![Parameters for the new Onto Root Gazetteer](image)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Required</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>caseSensitive</td>
<td>Boolean</td>
<td>✓</td>
<td>false</td>
</tr>
<tr>
<td>considerHeuristicRules</td>
<td>Boolean</td>
<td>✓</td>
<td>false</td>
</tr>
<tr>
<td>considerProperties</td>
<td>Boolean</td>
<td>✓</td>
<td>true</td>
</tr>
<tr>
<td>morpher</td>
<td>Morph</td>
<td>✓</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>ontology</td>
<td>ontology</td>
<td>✓</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>posTagger</td>
<td>POSTagger</td>
<td>✓</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>propertiesToExclude</td>
<td>String</td>
<td>✓</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>propertiesToInclude</td>
<td>String</td>
<td>✓</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>separateCamelCasedWords</td>
<td>Boolean</td>
<td>✓</td>
<td>true</td>
</tr>
<tr>
<td>tokeniser</td>
<td>DefaultTokeniser</td>
<td>✓</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>typesToConsider</td>
<td>Set</td>
<td></td>
<td>[]</td>
</tr>
<tr>
<td>useResourceUri</td>
<td>Boolean</td>
<td>✓</td>
<td>true</td>
</tr>
</tbody>
</table>

!!! Must add “class”, “instance”, “property” (bug in GATE 7.1, later: default)
Running the OntoRoot gazetteer

- If mostly matching proper names, then add to your application and run like the ANNIE gazetteer

- It will match against the document text as it is, which is not ideal if matching against terms (“leaders” should match “leader”: need lemma/root)

- To find root we need: Tokeniser, Sentence Splitter, POS tagger, and Morphological Analyser

- To match the root and not the text, use Flexible Gazettee PR with OntoRoot as the embedded gazetteer

- Flexible Gazettee delegates to OntoRoot Gazettee: Flexible Gazettee is the one that needs to be added to the application!
  → If Flexible Gazettee is used, no need to add OntoRoot Gazettee to application.
OntoRoot Application in GATE

Create a Flexible Gazetteer with an OntoRoot inside it

Build a GATE application with the PRs shown
Output Example

- The URI feature contains the matched class or instance URI
- The type feature is either class or instance
- Instances have also features classURI and classURIList
Hands-on 6: OntoRootGazetteer

• Load Gazetteer_Ontology_Based plugin, ANNIE and Tools plugins

• Close any open ontologies, but keep the document you have open
  Load the ontology test-ontology-instances.owl

• Create a new corpus pipeline

• Create Document Reset, Tokeniser, Sentence Splitter, POS Tagger, and Morphological Analyser (all with defaults) and add to the pipeline in that order

• Create separate Tokeniser, POS Tagger, and Morphological Analyser PRs for OntoRoot Gaz, name them OR-Tokeniser etc.

• Create and configure OntoRootGazetteer: chose ontology and make sure the OR-... tokeniser, POS Tagger, Morpher are selected

• Add “class”, “instance” and “property” to typesToConsider

• Continue on next page
Hands-on 6: OntoRoot (contd.)

- Create a FlexibleGazetteer PR:
  - add Token.root to inputFeatureNames
  - choose the OntoRoot gazetteer as gazetteerInst
- Add Flexible Gazetteer to the pipeline
- Set the runtime parameter setsToRemove of the Document Reset to “Test”
- Set all the input and output sets in the pipeline to Test
- Create a corpus for document voting-example.xml
- Run the pipeline and inspect the resulting Lookup annotations in the Test annotation set
- Save your application and keep it open for later
Conventions in GATE

- We use “Mention” annotations to reflect the fact that the text mentions a particular instance or a class.
- The Mention annotations have two special features:
  - $\textit{class} =$ class URI from the ontology
  - $\textit{inst} =$ instance URI from the ontology (if available)
    - e.g. Mention \{class=Leader, inst=Gordon_Brown\}
- It's important \textbf{not} to use $\textit{class}$ and $\textit{inst}$ as features unless you're dealing with ontologies, as these are predefined names in several tools.
- OntoRoot Gazetteer does not follow the conventions.
Compatibility with OntoRootGazetteer

- The OntoRootGazetteer always puts the matching resource (class or individual) URI in a feature called “URI” and the kind of match in a feature called “type”. For individuals it also creates the features “classURI” and “classURIList”

- But GATE/JAPE requires these features to be called **class** and **inst**

- So we need a JAPE grammar to first change the names of these features
Phase: LookupRename
Input: Lookup
Options: control = appelt

Rule: RenameLookup

{Lookup.type == instance}

:match

--> :match{
  for (Annotation lookup : matchAnnots) {
    FeatureMap theFeatures = lookup.getFeatures();
    theFeatures.put("class", theFeatures.get("classURI"));
    theFeatures.put("inst", theFeatures.get("URI"));
  }
}
Ontology Aware JAPE

- JAPE transducers have a run-time parameter which is an ontology
- [Note that the ANNIE NE Transducer] does not have this parameter, so you cannot use it for ontology-aware JAPE]
- By default it is left blank, so not used during LHS matching
- When an ontology is provided, the class feature can be used on the LHS of a JAPE rule
- When matching the class value, the ontology is checked for subsumption: any subclass on the left side of “==” matches
- e.g. {Lookup.class == Person} will match a Lookup annotation with class feature, whose value is either Person or any subclass of it
Ontology-aware JAPE example

Phase: OntoMatching
Input: Lookup
Options: control = appelt

Rule: PersonLookup

\[
\{\text{Lookup.class} == \text{Person}\}
\]

\rightarrow

\{\text{class} = \text{:person.Lookup.class},
\text{inst} = \text{:person.Lookup.inst}\}

Matches the class Person or any of its subclasses

Adds class and instance information as features on the Mention annotation
Ontology-aware JAPE example

Ontology-aware JAPE applies only to a feature named “class” and only if the PR's ontology parameter is set.

```
{Lookup.class == "http://example.com/stuff#Person"}

Matches this class or any subclass in the ontology
```

```
{Lookup.class == "Person"}

If the string is not a full URI, JAPE adds the default namespace from the ontology, looks up that class in the ontology, and matches it or any subclasses. Be very careful if your ontology uses more than one namespace!
```

These rules apply equally to the string in the JAPE rule and in the value of the annotation's class feature.
Template declarations can be used to simplify namespaces.

Template: protont =
   “http://proton.semanticweb.org/2005/04/protont#${n}”
...
{Lookup.class == [protont n=Person]}
...
{Lookup.class == [protont n=Location]}

If you switch to a newer version of PROTON, you only need to change the Template declarations, not every JAPE LHS. (See the GATE User Guide http://gate.ac.uk/userguide/sec:jape:templates for more details and examples.)

Template: protont =
   “http://proton.semanticweb.org/2006/05/protont#${n}”
...
Matching subclasses

David Cameron was the first of the main UK party leaders...

The rule matches because Leader is a subclass of Person
Hands-on 7: ontology-aware JAPE

- Load the JAPE transducer `rename-lookup-features.jape` and add to the end of your existing pipeline
  Set the input and output sets for it to Test
- Run the modified pipeline to see how the Lookup annotations for individuals in Test now have class features
- Load the JAPE transducer `person-onto-matching.jape` and add it to the end of the pipeline as before.
  - Set the input and output sets for it to Test
  - Select the ontology as the run-time param
- Run the modified pipeline to see how it creates new Mention annotations
- Save the application with a new name and close it
The LKB gazetteer is used to do ontology-based gazetteer lookup against very large ontologies, e.g. DBPedia, GeoNames and other Open Linked Data ontologies.

Uses a SPARQL query to create a gazetteer list from the ontology:

```sql
SELECT DISTINCT ?label ?inst ?class
WHERE {
  FILTER (lang(?label) = "en")
}
```

Internally retrieves the result rows and converts them to gazetteer entries with inst and class features.

Creates a cache file that will load fast subsequently.
LKB: Continued

- Lives in plugin Gazetteer_LKB
- LKB does not use the GATE ontology language resources. Instead, it uses its own mechanism to load and process ontologies.

- Set up your dictionary first. The dictionary is a folder with some configuration files. Use the samples at GATE_HOME/plugins/Gazetteer_LKB/samples as a guide or download a pre-built dictionary from ontotext.com/kim/lkb_gazetteer/dictionaries.

- The dictionary directory defines which repository to connect to, which SPARQL queries to use to initialise the gazetteer, etc.

- For details see http://gate.ac.uk/userguide/sec:gazetteers:lkb-gazetteer
Other Gazetteers

- Often ontologies are huge
  → need gazetteers that can deal with very large gazetteer lists, do not want to re-create list too often
- Often we need to use specific SPARQL queries, need to process/clean labels or property values before using for the gazetteer

=> Separate preprocessing pipeline to create large gazetteer files with inst and class features

- Use a gazetteer that can handle large files:
  LKB Gazetteer with list files (not SPARQL):
    GATE version > 7.1 can handle class and inst features
  ExtendedGazetteer from StringAnnotation plugin
    (http://code.google.com/p/gateplugin-stringannotation/)
    can handle arbitrary features
Semantic Enrichment

- Add additional knowledge to semantically annotated mentions
- Simplest: add features
  - e.g. add the name of the country, zip code for a city
  - if we have city names to disambiguate, may use zip code to disambiguate!
- Use Java API in JAPE RHS, Groovy or own PR
- SemanticEnrichment PR from the Gazetteer_LKB plugin
  - SPARQL Endpoint (not GATE Ontology LR)
  - Run SPARQL query for each URI in inst
  - add query result to 'connections' feature
The Big Picture

Ontology

SPARQL, Filtering
GATE processing
Format Conversion

Corpus

Gazetter
LST Files

Linguistic Processing: Tokens, POS, Corefs ...

Semantic Annotation, Enrichment

Disambiguation, Postprocessing

O-Population

Mimir PR

???

New Ontology

Mimir Server
GATE Mímir

- Server-based, index large numbers of GATE documents
  - Text (Tokens)
  - Annotations and their features
  - Semantics: links to ontologies

- Can combine any of these into complex queries
  SPARQL can be used to semantic annotations based on the ontology:

\[
\text{(European Union) \&}
\]
\[
\{ \text{Person sparql = ”SELECT ?inst \{} \\
\text{ ?inst :partyMember :LabourPartyUK .} \\
\text{ ?inst :birthPlace ?x .} \\
\text{ ?x :locatedIn :Wales . } \text{”} \}
\]
Performance Evaluation

• Mention annotations can be evaluated against a gold standard by matching the classes or instances

• However, traditional IE evaluation measures (Precision and Recall) don't take into account the class hierarchy

• Some mistakes can be “more wrong” than others
  • Nick Clegg → Person (not Leader) – still logically correct
  • Nick Clegg → Location – wrong

• We need a way of dealing with this, to give some credit for these kind of situations
Balanced Distance Metric

- BDM measures the closeness of two concepts in an ontology or taxonomy
- It produces a real number between 0 and 1
- The more closely related the two concepts are in an ontology, the greater their BDM score is
- It is dependent on a number of features:
  - the length of the shortest path connecting the two concepts
  - the depth of the two concepts in the ontology
  - size and density of the ontology
1: Use OAT to create gold standard

By convention, change the OAT default to put the annotations in the Key set. It is already configured to create Mentions with class and inst features.
2: Compute BDM

- Located in the Ontology_BDM_Computation plugin
- Can be run in a non-corpus pipeline
  Runtime parameters:
  - input ontology
  - output txt file
- For each pair of classes in the ontology, it calculates a number of statistics
- Since BDM is symmetric for any two concepts, the resulting file contains only one entry per pair, despite one being called key and the other response.

Example file: module-9-hands-on/bonus/bdm-output.txt

key=http://gate.ac.uk/example#Entity, response=http://gate.ac.uk/example#Location, bdm=0.0, msca=http://gate.ac.uk/example#Entity, cp=0, dpk=0, dpr=1, n0=1.6666666, n1=1.6666666, n2=2.0, bran=1.8000001
3: Calculate BDM-aware measures

- The IAA plugin computes precision, recall, and F-measure over a corpus.
  BDM statistics file can optionally be used to make BDM-aware.
- Corpus Quality Assurance VR for a corpus can calculate and show precision, recall and F-measure (strict+lenient) over a corpus and for each document in a corpus.
  BDM statistics file can optionally be used to calculate BDM-aware and traditional measures.
Our example text again

David Cameron was the first of the main UK party leaders to cast their vote. The Tory leader went to a community hall in Witney, Oxfordshire, shortly after 1030 BST, accompanied by his wife Samantha.

Labour leader Gordon Brown went to vote shortly after 1100 BST at a community centre close to his home in North Queensferry, Fife. His wife Sarah was with him.

Nick Clegg, leader of the Liberal Democrats, arrived at a polling station in Sheffield Hallam at 1120 BST. His wife Miriam is unable to vote in the general election because she is a Spanish citizen.

The leader of the Scottish National Party, Alex Salmond, cast his vote shortly before noon, at Macduff in Banffshire. Leuan Wyn Jones of Plaid Cymru voted in the constituency of Ynys Mon in north Wales at lunchtime.

Clegg is marked as a Person, instead of Leader

Salmond is missing
Results

- Traditional scores for the example:
  - Match = 2, Only A (missing) = 2, Only B (spurious) = 1, Overlap (Partial) = 0
  - Recall = 0.50, Precision = 0.67, F1 = 0.57

- BDM-sensitive scores:
  - Recall = 0.60, Precision = 0.81, F1 = 0.69
Further materials

Ontology design:
principles: http://lsdis.cs.uga.edu/SemWebCourse/OntologyDesign.ppt

BDM: http://gate.ac.uk/userguide/sec:eval:bdmplugin

Semantic Annotation:


QUESTIONS?