Introduction to GATE

Andrés García-Silva, Elena Montiel-Ponsoda

Ontology Engineering Group
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo s/n
28660 Boadilla del Monte, Madrid, Spain

http://www.oeg-upm.net

emontiel@fi.upm.es
hagarcia@delicias.dia.fi.upm.es
Phone: 34.91.3363670
Fax: 34.91.3524819

Date: December 10th, 2009

Outline

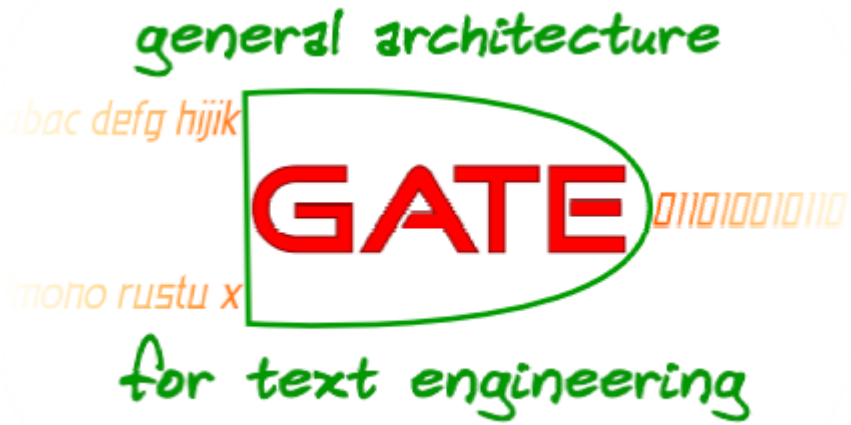
- Introduction to GATE
- ANNIE
  - ANNIE Demo
- Jape Rules
  - Jape Rules Demo
- Machine Learning
- Ontology support
  - Semantic Annotation
  - Ontology Learning
  - Semantic Annotation Demo
- LSPs Demo

Outline

- **Introduction to GATE**
- ANNIE
  - ANNIE Demo
- Jape Rules
  - Jape Rules Demo
- Machine Learning
- Ontology support
  - Semantic Annotation
  - Ontology Learning
  - Semantic Annotation Demo
- LSPs Demo

Introduction to GATE

- **GATE** stands for: **G**eneral **A**rchitecture for **T**ext **E**ngineering

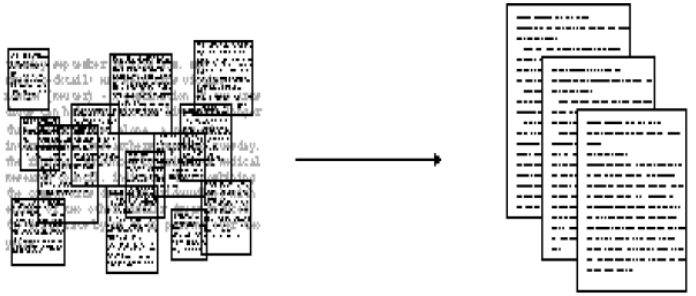


- It is an infrastructure for developing and deploying software components that process human language
- GATE has been in development at the University of Sheffield since 1995

Introduction to GATE

- **GATE** was originally developed as an Information Extraction system (IE)
  - What does an IE system do?
  - Which is the difference between IE and IR (Information Retrieval)?

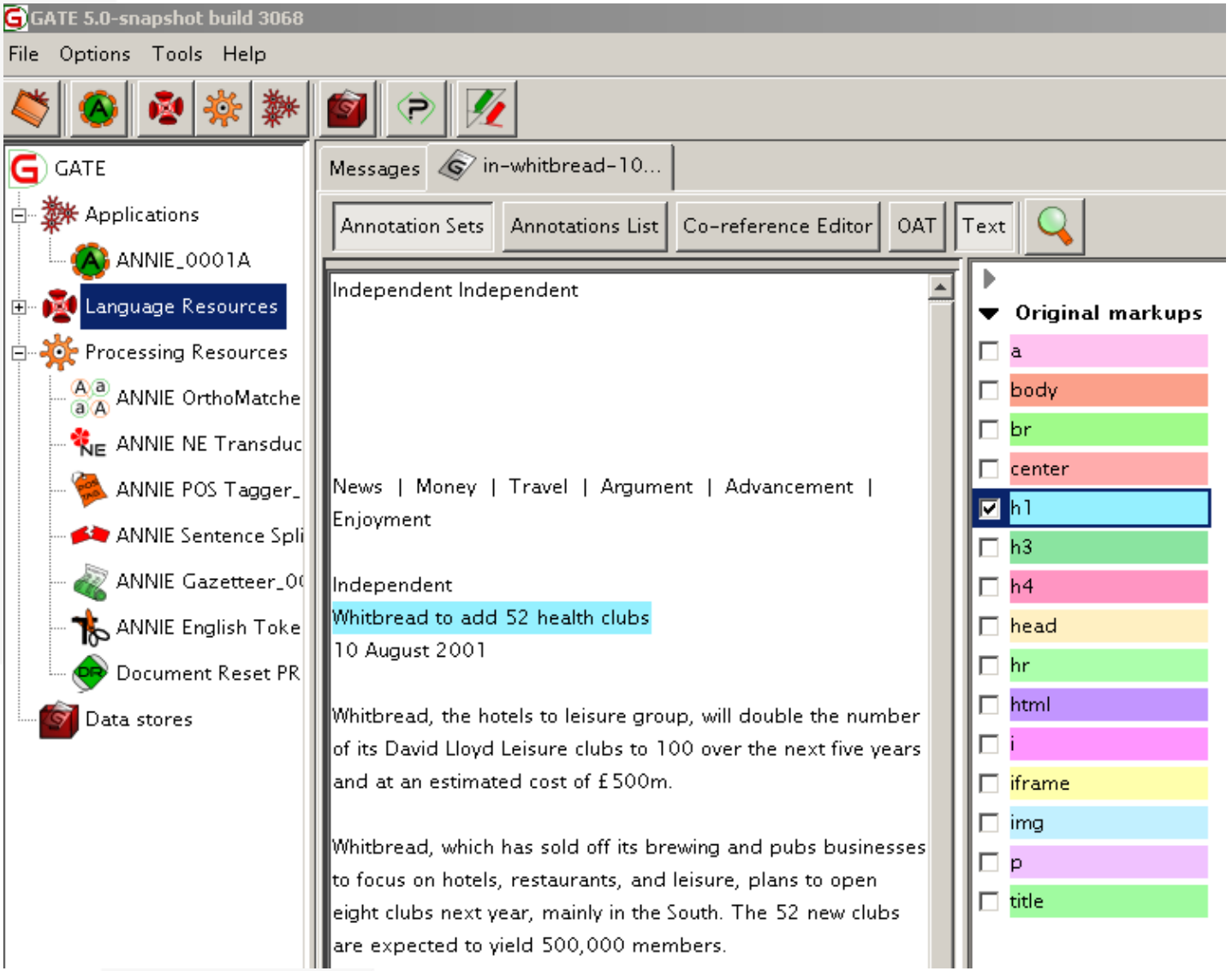
IR pulls **documents** from large text collections (usually the Web) in response to specific keywords or queries. You analyse the **documents**.



IE pulls **facts** and **structured information** from the content of large text collections. You analyse the **facts**.

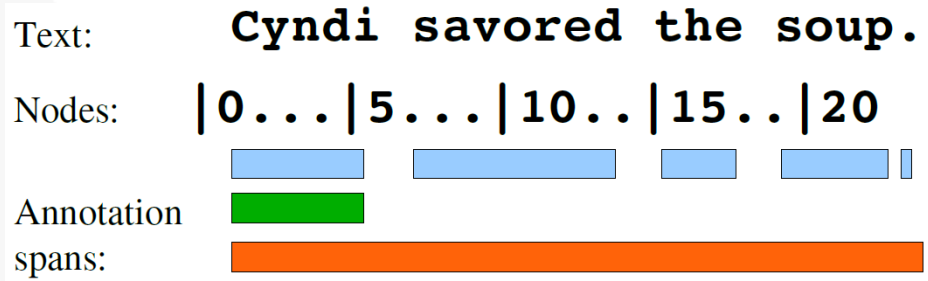
- GATE integrates 3 types of resources:
  - **Language Resources (LRs) or DATA**, e.g. corpora, gazetteer, ontologies
  - **Processing Resources (PRs) or ALGORITHMS**, e.g. lemmatisers, parsers, taggers
  - **Visual Resources (VRs) or GUI**, i.e. visualization and editing components
- Algorithms + Data + GUI = **APPLICATIONS**

Introduction to GATE



What are annotations?

- Annotations are linguistic information or metadata associated to the content in documents
- Annotations consist of features with a name and a value. E.g.: dog **TOKEN kind=word**
  pos=NN (Part of speech)
  length=3



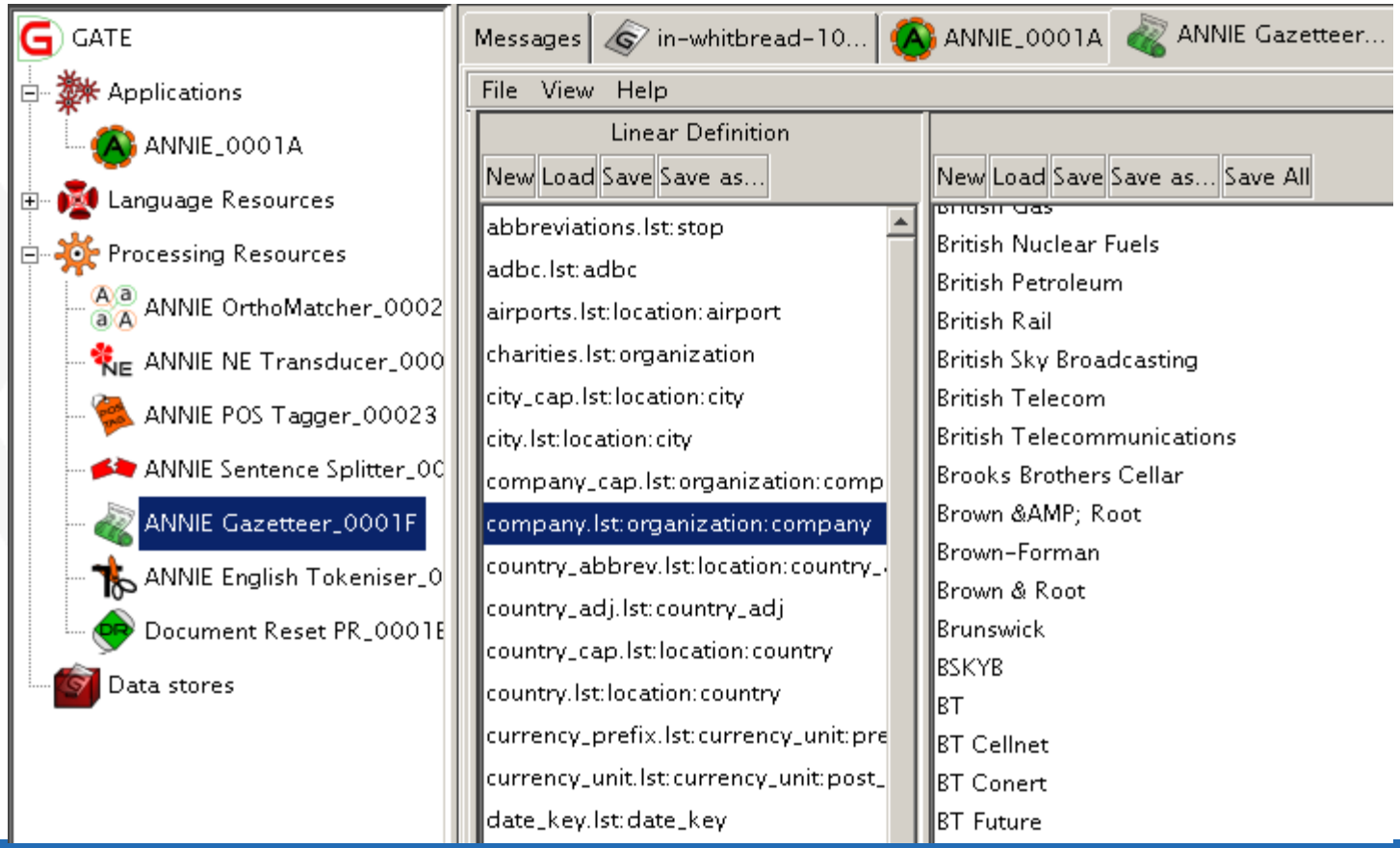
Annotation descriptions:

Id	Type	Start	End	Features
1	token	0	5	pos=NP
2	token	6	13	pos=VBD
3	token	14	17	pos=DT
4	token	18	22	pos=NN
5	token	22	23	
6	name	0	5	type=person
7	sentence	0	23	

- **NP: proper noun**
- **VBD: verb past tense**
- **DT: determiner**
- **NN: noun-singular**

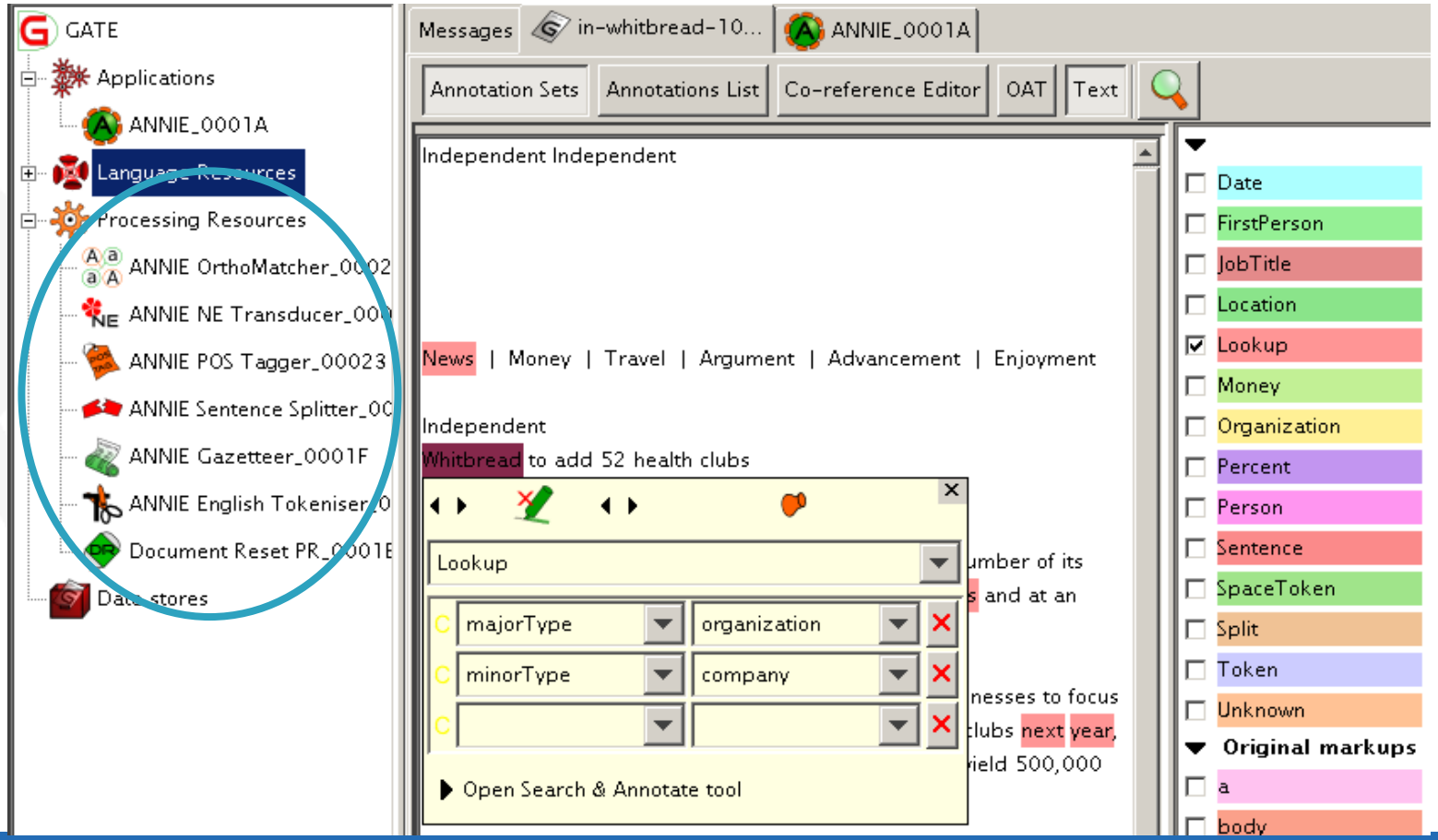
What are Lexical Resources?

- Lexical Resources can be corpora, gazetteer, ontologies…
- Gazetteers are plain lists of entities grouped into categories.
  - E.g.: Location - European capital cities: Paris, Madrid, Berlin, etc.
  - (**Gazetteer PR** to conduct Named Entity or key phrase Lookup)

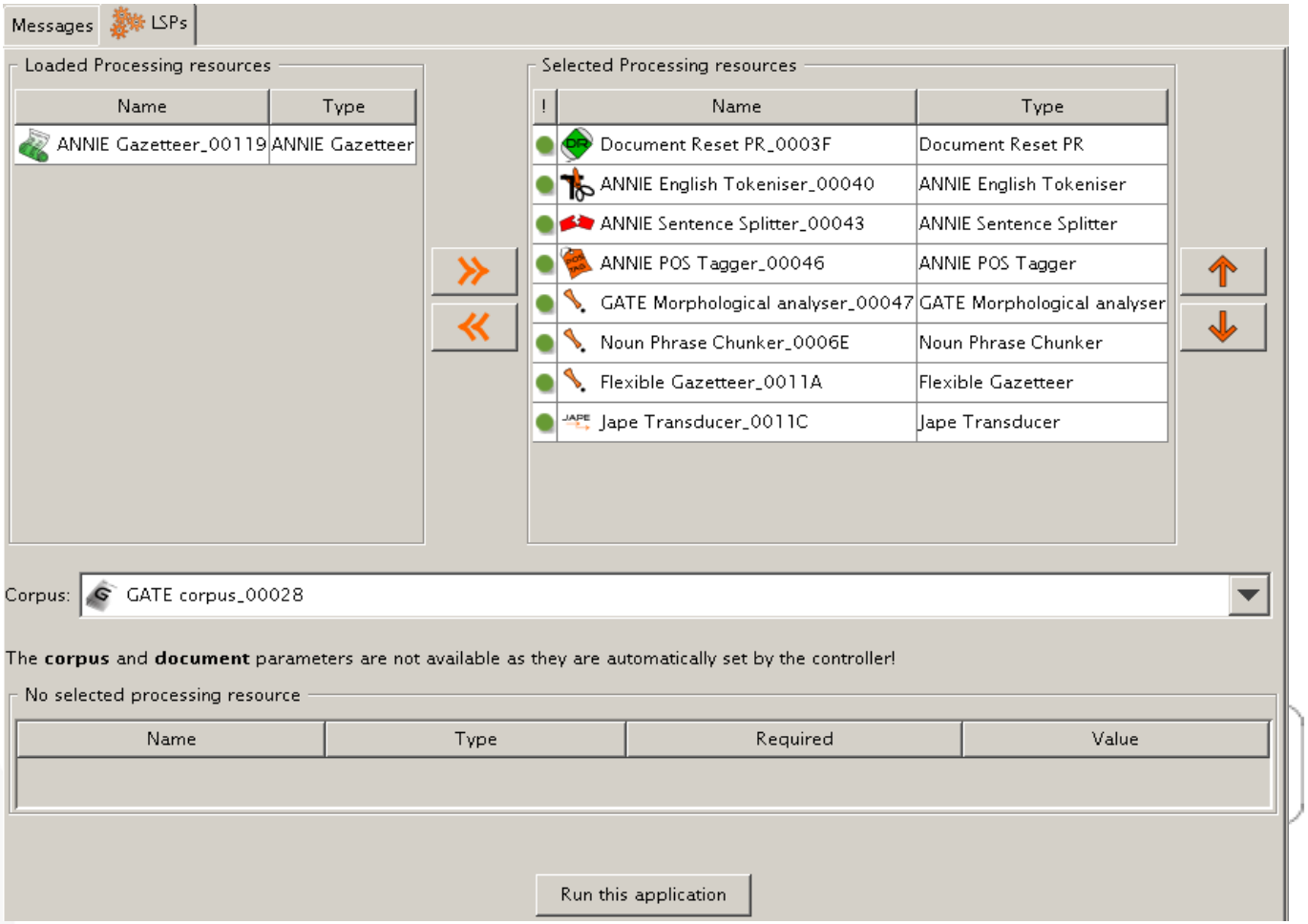


What are Processing Resources?

- PRs take as input LRs and produce as output **annotations** on those LRs
- E.g.: tokeniser, pos tagger, gazetteer, orthomatcher...



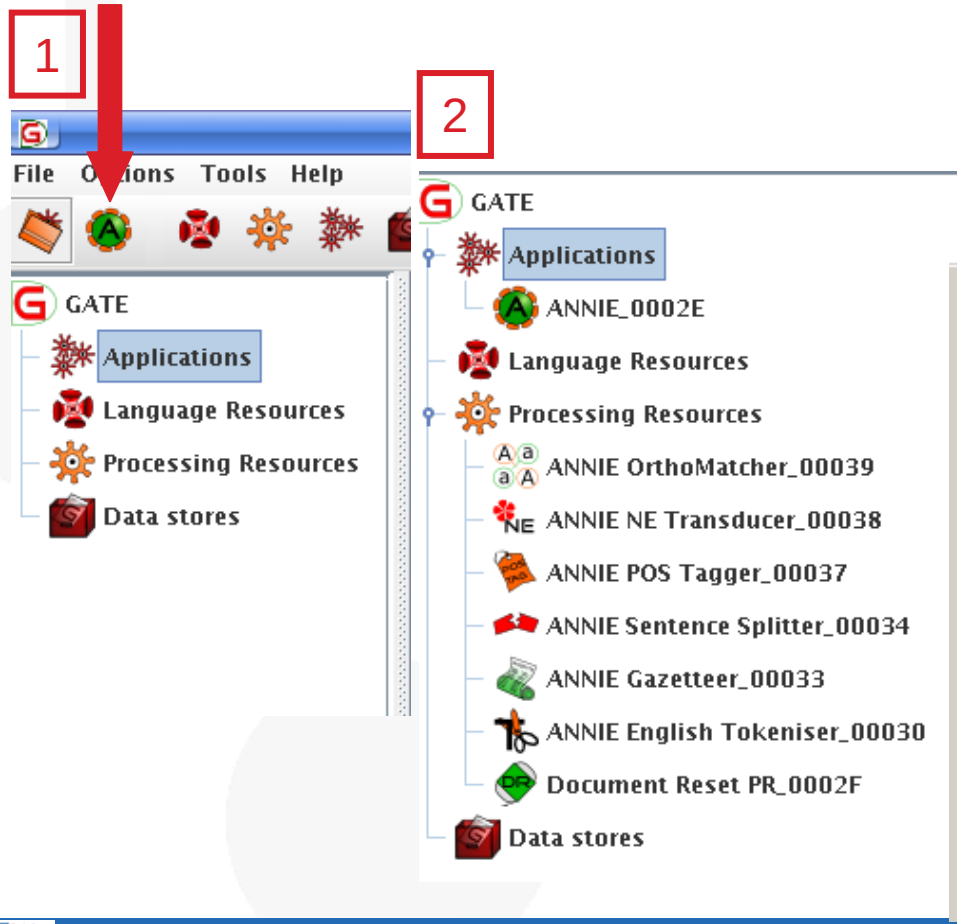
- **Applications** execute PRs in a particular order
- Currently only sequential, or pipeline, execution is supported:

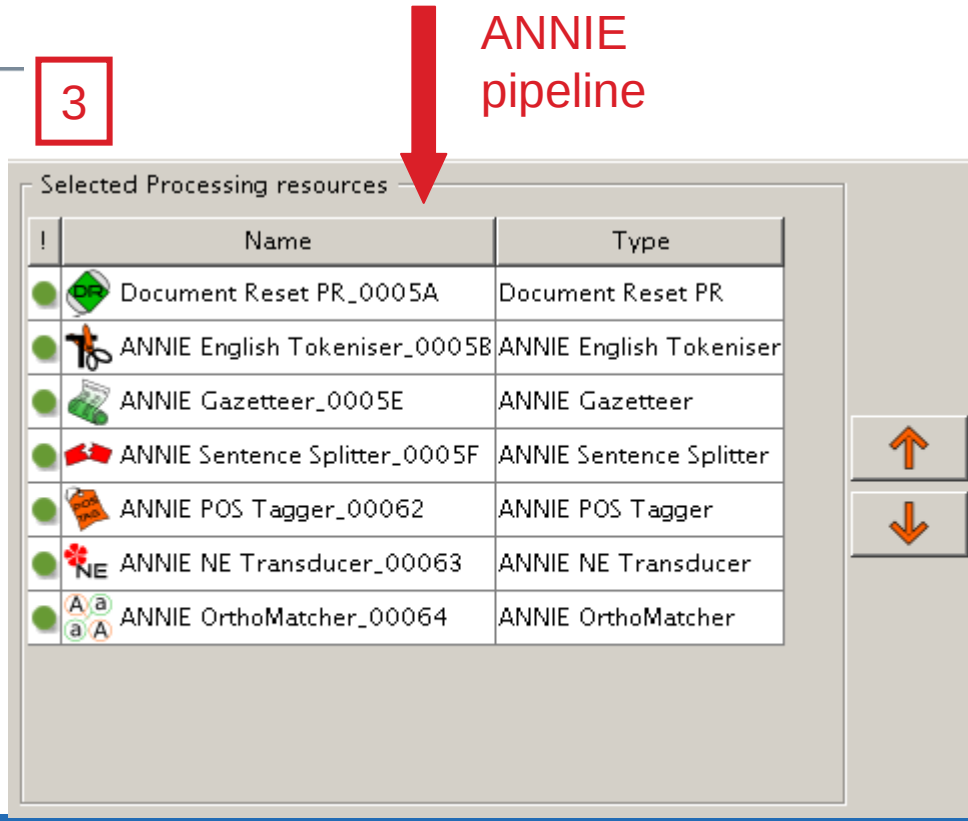


Outline

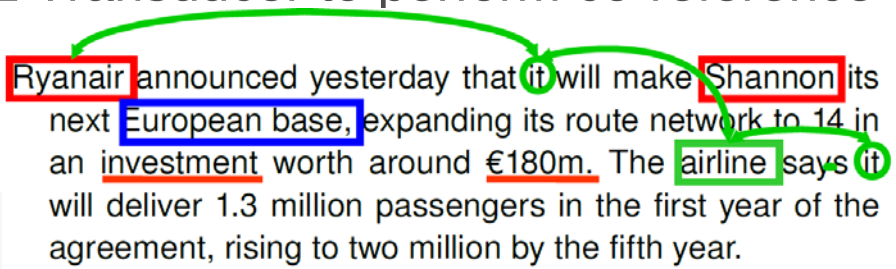
- Introduction to GATE
- **ANNIE**
  - ANNIE Demo
- Jape Rules
  - Jape Rules Demo
- Machine Learning
- Ontology support
  - Semantic Annotation
  - Ontology Learning
    - Ontology Learning Demo
- LSPs Demo

- ANNIE is GATE's basic building block
- ANNIE consists of several PRs that performe annotations (which range from basic annotations to more complex ones)
- The PRs in ANNIE are enough to create an NLP application





- Document Reset: clears existing annotations in the document (no annotation is embedded in the document)
- ANNIE English Tokeniser: splits the text into tokens
- ANNIE Gazetteer: wordlists grouped in different categories to perform NE or Key Phrase Lookups
- ANNIE Sentence Splitter: assigns annotations to sentence boundaries
- ANNIE PoS Tagger: assigns PoS taggs to tokens
- ANNIE Orthomatcher: adds identity relations between NE found by the NE Transducer to perform co-reference



- ANNIE NE Transducers: executes JAPE rules to create complex annotations based on the results of the previous PRs

Outline

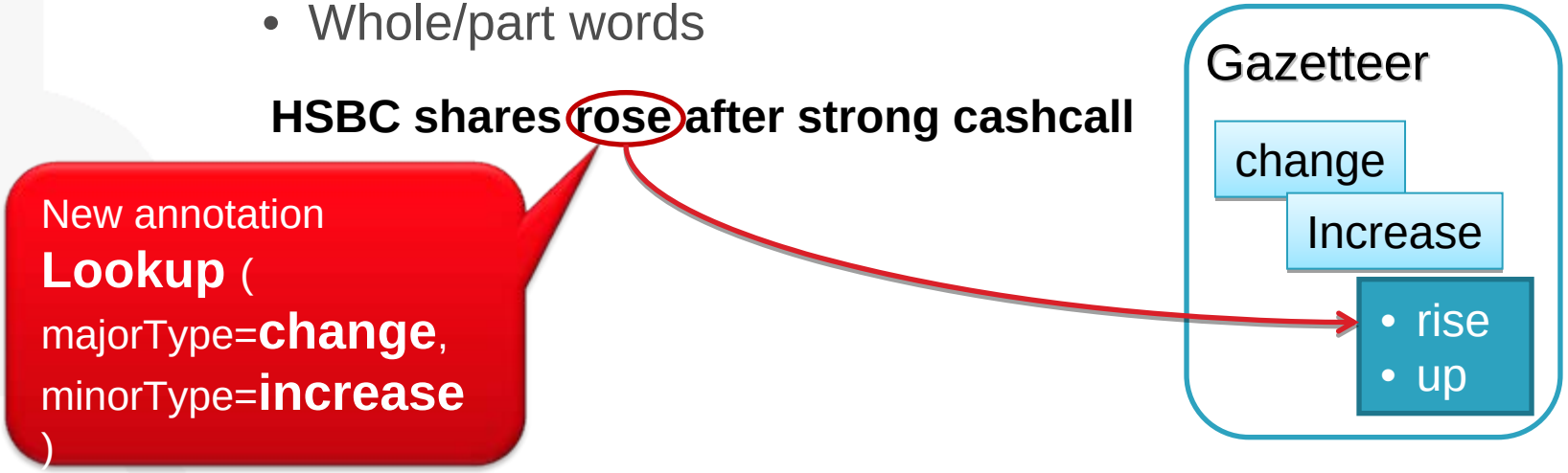
- Introduction to GATE
- ANNIE
  - **ANNIE Demo**
- Jape Rules
  - Jape Rules Demo
- Machine Learning
- Ontology support
  - Semantic Annotation
  - Ontology Learning
  - Semantic annotation Demo
- LSPs Demo

Outline

- Introduction to GATE
- ANNIE
  - ANNIE Demo
- **Jape Rules**
  - Jape Rules Demo
- Machine Learning
- Ontology support
  - Semantic Annotation
  - Ontology Learning
  - Semantic Annotation Demo
- LSPs Demo

Limitations of Gazetteer Lists

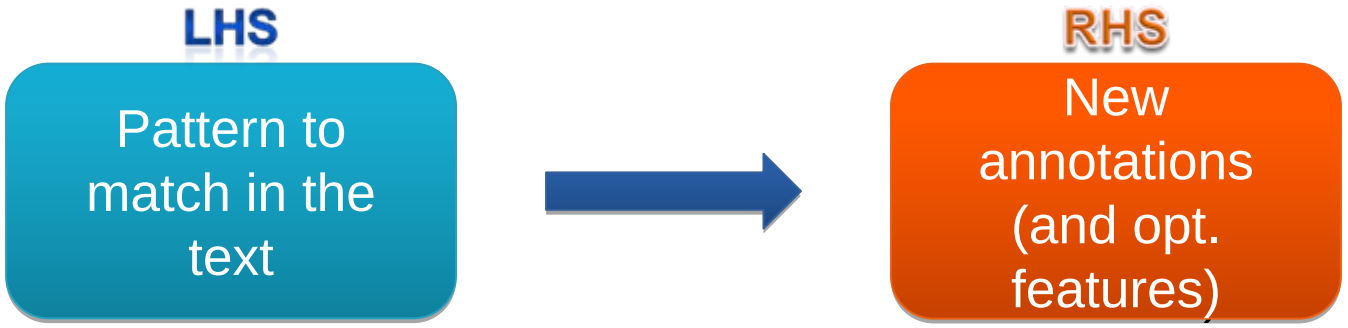
- Gazetteer lists are designed for annotating simple, regular features
  - Some flexibility is provided by matching
    - Word roots
    - Whole/part words



recognizing e-mail addresses using just a gazetteer would be impossible

What is JAPE?

- JAPE provides pattern matching in GATE



"contact information: JohnMalkovich@gmail.com

({**Token**.kind == word})*
{**Token**.string == "@"}
({**Token**.kind == word})*
({**Token**.string == "."}
({**Token**.kind == word})
:**email**

New annotation
Email (
rule = "emailaddress")

Email =
emailaddress"}

- JAPE rules combine to create a phase
- Phases combine to create a grammar

JAPE RULES

Jape rule Example

```
Phase: Email
Input: Token SpaceToken
Options: control = appelt

Macro: WORD_OR_NUMBER
(
({Token.kind == word}|{Token.kind == number})
)

Rule: emailaddress
Priority: 50
(
(WORD_OR_NUMBER)+
({Token.string == "."}(WORD_OR_NUMBER)+)*
{Token.string == "@"}
(WORD_OR_NUMBER)+
({Token.string == "."}(WORD_OR_NUMBER)+)*
)
:email -->
:email.EMail= {rule = "emailaddress"}
```

Left Hand Side Patterns

LHS

```
({Token.kind == word})*
{Token.string == "@"}
({Token.kind == word})*
({Token.string == "."}
({Token.kind == word})
:email
```

- LHS:
  - Annotations
  - Optionally features and their values
- Any annotation to be used must be included in the input header
- Each annotation is enclosed in curly braces
- Annotations may be combined using traditional Klene operators: | * + ?
- Each pattern to be matched is enclosed in round brackets and can have a label attached

Left Hand Side Patterns

LHS

```
({Token.kind == word})*
{Token.string == "@"}
({Token.kind == word})*
({Token.string == "."}
({Token.kind == word})
:email
```

- JAPE rules can access annotation features
- Features can be compared with ==, !=, >,<, =~, !~, ==~ and !=~
- Ranges can be specified({Token})[1,3] or ({Token})[3]

- Contextual information can be specified in the same way, but has no label
- Contextual information will be consumed by the rule

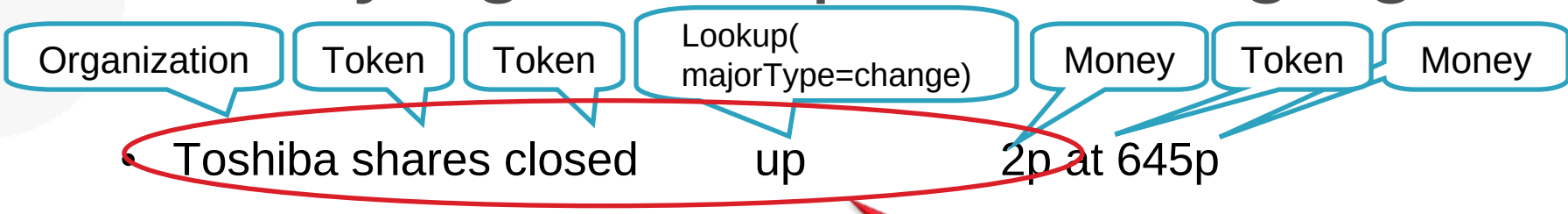
  ({Annotation1})
  ({Annotation2}):match
  ({Annotation3})

Right Hand Side Annotations

- LHS and RHS are separated by -->
- Label matches that on the LHS
- Annotation to be created follows the label

(Annotation1):**label -->**
:label.NewAnnotation =
{feature1 = value1,
feature2 = value2}

- Identifying share price changing:




```
Phase: Shares
Input: Token Organization Lookup Money
Percent
Options: control = appelt

Rule:ShareChange
(
        {Organization}
        ({Token})[0,3]
        {Lookup.majorType=="change"}
        ({Token})[0,3]
        ({Money}|{Percent})
):change -->
:change.ShareChange = {rule =
"ShareChange"}
```

New annotation
ShareChange(
rule = “ShareChange”)

Outline

- Introduction to GATE
- ANNIE
  - ANNIE Demo
- Jape Rules
  - **Jape Rules Demo**
- Machine Learning
- Ontology support
  - Semantic Annotation
  - Ontology Learning
  - Semantic Annotation Demo
- LSPs Demo

- Corpus creation
- Default ANNIE application
- Gazetteer recognizing changes
- Run ANNIE app
- Jape transducer loading our jape rule.
- Run ANNIE App to review the results.

Simple RHS Limitations

- The simple RHS of a JAPE rule can only add simple annotations and features
- Feature values are hard coded or can be copied from annotations matched by the LHS
- You may need more complex processing
  - Removing temporary annotations
  - Building complex features

RHS

```
:email.Email =
 {rule = "emailaddress"}
```

- RHS of a rule can consist of arbitrary Java code

JAPE RULES

JAVA Right Hand Side

- public void doit(Document doc, Map **bindings**, AnnotationSet annotations, AnnotationSet inputAS, AnnotationSet **outputAS**, Ontology ontology) throws JapeException
- Each labeled section of the LHS results in an Annotation Set that can be retrieved from the bindings map
  - AnnotationSet set = (AnnotationSet)bindings.get("labelname");
- All features of an annotation are stored in a map
  - FeatureMap map = annotation.getFeatures()
- Each feature is accessed by name
  - Object obj = map.get(“featurename”)
- New annotations should always be created in the outputAS
  - outputAS.add(start,end,label,features)

JAPE RULES

Java Right Hand Side Example

```
Rule:SC1_1
(
(LIST):subclass
{Lookup.minorType == be}
({Token.category == DT})?
{Lookup.majorType == CN}
({NounChunk}):superclass
)
-->
:superclass.Superclass = {rule="SC1_1"},
{
   // "subclass" matches LHS label
   List annList = new ArrayList((AnnotationSet)bindings.get("subclass"));
   //sort the list by offset
   Collections.sort(annList, new OffsetComparator());
   //iterate through the matched annotations
   for(int i = 0; i < annList.size(); i++)
   {
         Annotation anAnn = (Annotation)annList.get(i);
        // check that the new annotation is a NounChunk
        if ((anAnn.getType().equals("NounChunk")) )
        {
                    FeatureMap features = Factory.newFeatureMap();
                    // change this for a different rule name"
                    features.put("rule", "SC1_1");
                    // change "Subclass" for a different annotation name
                     annotations.add(anAnn.getStartNode(), anAnn.getEndNode(), "Subclass", features);
        }
    }
}
```

LSP: (NP<subclass>,)* and] NP<subclass> be [CN] NP<superclass>

Example: **Odometry, speedometry and GPS are types of sensors**.

NP: Noun Phrase
CN: Class Name

Outline

- Introduction to GATE
- ANNIE
  - ANNIE Demo
- Jape Rules
  - Jape Rules Demo
- **Machine Learning**
- Ontology support
  - Semantic Annotation
  - Ontology Learning
  - Semantic Annotation Demo
- LSPs Demo

Existing classified instances

- Typical supervised machine learning process (Classification):
  - We have items/instances and attributes/features
    - Car id=0001, brand=Ford, price=20000€, color=blue
    - Car id=0002, brand=Lexus, price=50000€, color=silver
  - We have classes: Car id=0001 -> Category = Sport
    Car id = 0002 -> Category = Luxury
  - Machine learning algorithm learns a relationship between the attributes and classes

New instance

- Car id=003, brand=Citroen, price= 5000€, color=orange, category=?

Machine Learning

Supervised classification

Training data set
(Classified instances)

New Instances
Unclassified

Instances (attributes,class)

Instance(brand=Citroen,
price= 5000€, color=orange)

Classifier

Category=familiar

Instances (attributes, class)

Evaluation data set
(Classified instances)

Machine Learning — Supervised classification

Token.String="California"
Lookup.majorType="Location"

Lookup.majorType="Job"

Entity.type=?

Token.category="Verb"

"California Governor Schwarzenegger proposes deep cuts"

Car example

- Instances
  - Cars
- Attributes
  - Color, Brand, Prize
- Class
  - Category

Annotations in Gate

- Instances: Annotations
  - Token, Lookup, Organization
- Attributes: Annotation Features
  - Token.string, Token. Root, Token.category(POS), Lookup.majorType, etc.
- Class: Annotation Feature
  - Entity.type = Person?

Machine Learning Gate – Supervised classification

Training data set
(Documents annotated with
The annotation feature we
want to learn)

Token.String="California"
Lookup.majorType="Location
"

Lookup.majorType
="Job"

Token.category="Verb
"

California Governor Schwarzenegger proposes…

New Instance
Unclassified

Docs(AnnotationFeatures
,class – entity.type)

Doc (Annotation.features)

Classifier

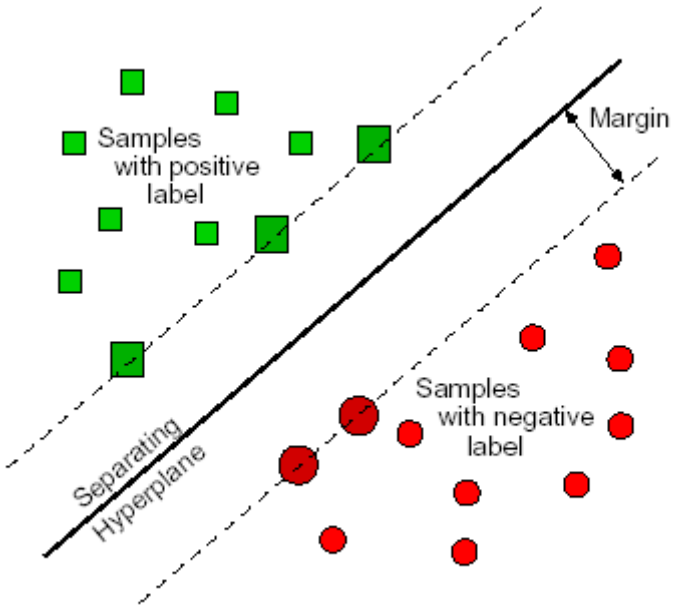
Schwarzenegger

Entity.type
=Person

Instances
(AnnotationFeatures,
class –entity.type)

Evaluation data set
(Documents annotated with
The annotation feature we want to learn)

- Attempt to find a hyperplane that separates data
- Goal: maximize margin separating two classes
- Wider margin = greater generalisation



Machine Learning

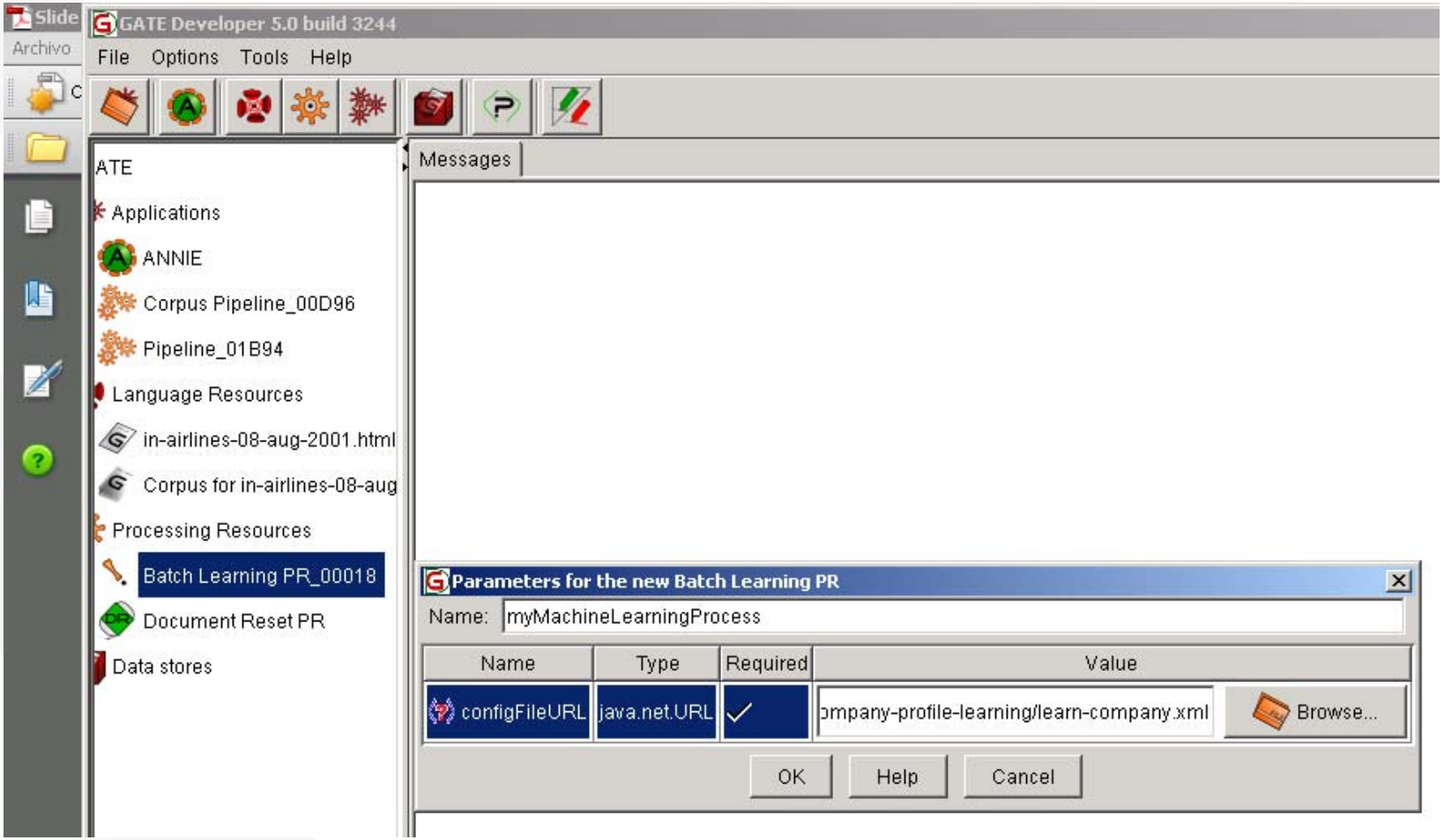
Configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<ML-CONFIG>
    <EVALUATION method="split" runs="2"
    ratio="0.66"/>
    <ENGINE nickname="SVM"
    implementationName="SVMLibSvmJava" options="
    -c 0.7 -t 0 -m 100 -tau 1  "/>

  <DATASET>
    <INSTANCE-TYPE>Token</INSTANCE-TYPE>
    <ATTRIBUTELIST>
            <NAME>Form</NAME>
            <SEMTYPE>NOMINAL</SEMTYPE>
            <TYPE>Token</TYPE>
            <FEATURE>string</FEATURE>
            <RANGE from="-5" to="5"/>
    </ATTRIBUTELIST>
    <ATTRIBUTELIST>
            <NAME>category</NAME>
            <SEMTYPE>NOMINAL</SEMTYPE>
            <TYPE>Token</TYPE>
            <FEATURE>category</FEATURE>
            <RANGE from="-5" to="5"/>
    </ATTRIBUTELIST>
    <ATTRIBUTELIST>
            <NAME>Tokenkind</NAME>
            <SEMTYPE>NOMINAL</SEMTYPE>
            <TYPE>Token</TYPE>
            <FEATURE>kind</FEATURE>
            <RANGE from="-5" to="5"/>
    </ATTRIBUTELIST>
    <ATTRIBUTELIST>
            <NAME>Lemma</NAME>
            <SEMTYPE>NOMINAL</SEMTYPE>
            <TYPE>Token</TYPE>
            <FEATURE>root</FEATURE>
            <RANGE from="-5" to="5"/>
    </ATTRIBUTELIST>
    <ATTRIBUTELIST>
            <NAME>Gaz</NAME>
            <SEMTYPE>NOMINAL</SEMTYPE>
            <TYPE>Lookup</TYPE>
            <FEATURE>majorType</FEATURE>
            <RANGE from="-5" to="5"/>
    </ATTRIBUTELIST>
    <ATTRIBUTE>
                <NAME>ENTITY</NAME>
                <SEMTYPE>NOMINAL</SEMTYPE>
                <TYPE>Entity</TYPE>
                <FEATURE>type</FEATURE>
                <POSITION>0</POSITION>
     </ATTRIBUTE>
    <ATTRIBUTE>
            <NAME>Class</NAME>
            <SEMTYPE>NOMINAL</SEMTYPE>
            <TYPE>Mention</TYPE>
            <FEATURE>class</FEATURE>
            <POSITION>0</POSITION>
            <CLASS/>
    </ATTRIBUTE>
     </DATASET>
</ML_CONFIG>
```

Machine Learning

Use in Gate



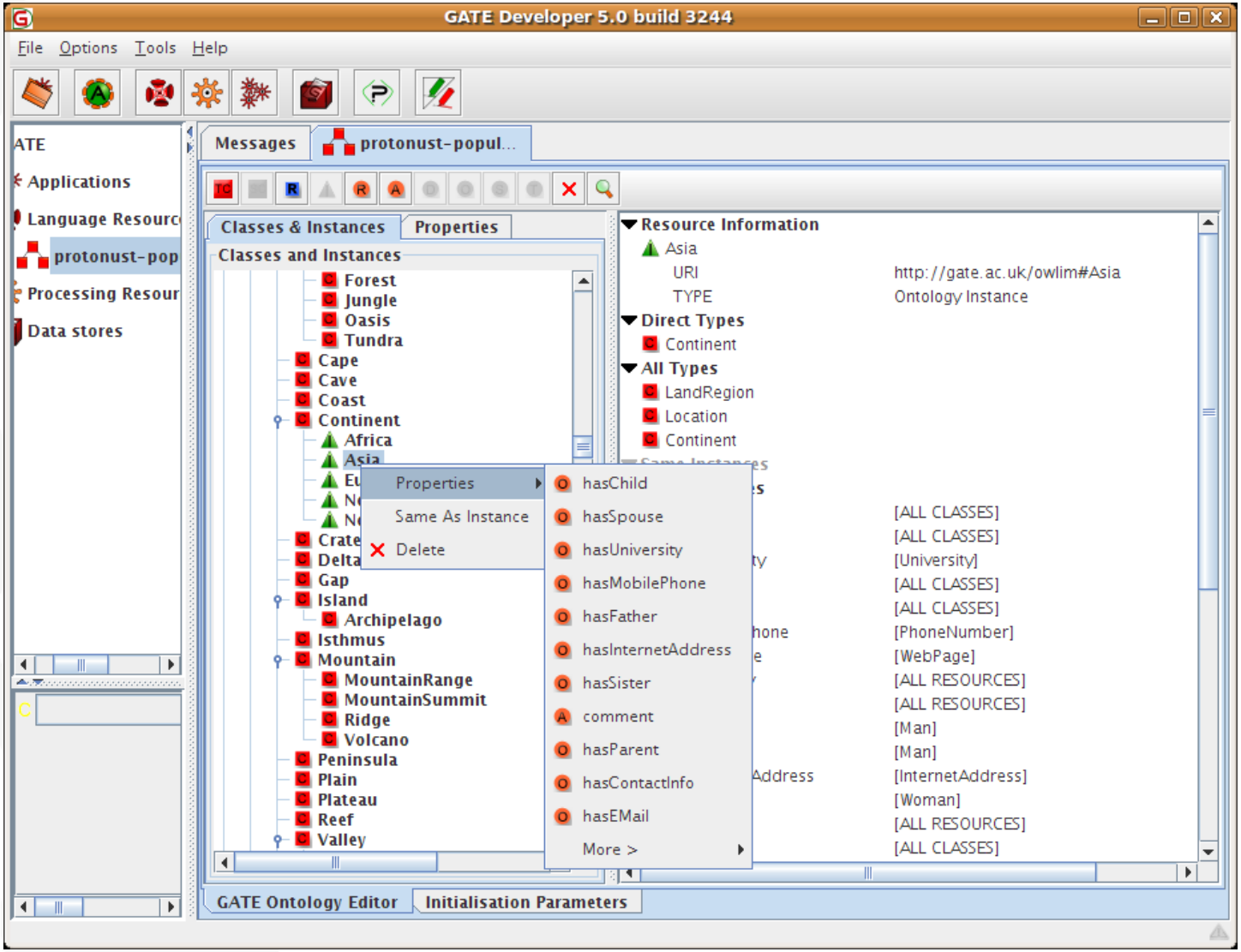
Outline

- Introduction to GATE
- ANNIE
  - ANNIE Demo
- Jape Rules
  - Jape Rules Demo
- Machine Learning
- **Ontology support**
  - Semantic Annotation
  - Ontology Learning
  - Semantic Annotation Demo
- LSPs Demo

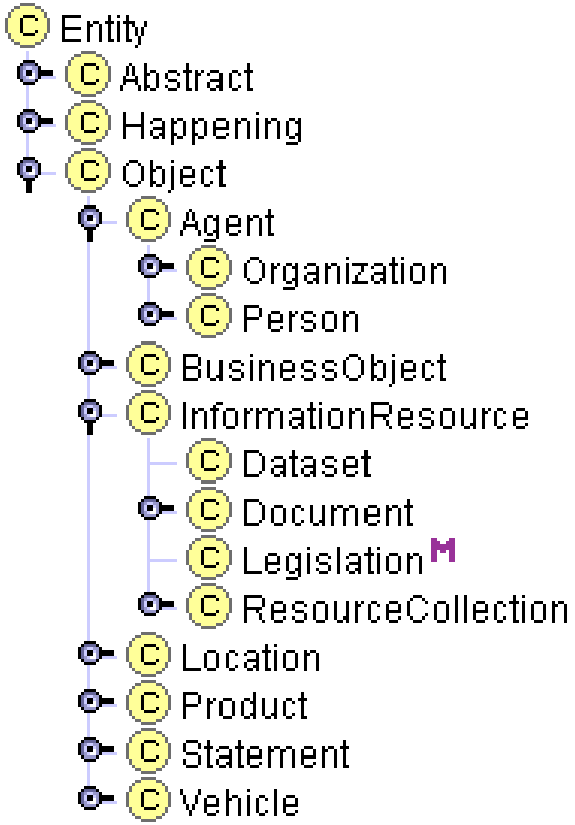
- Comes with one concrete implementation
  - preinstalled: Sesame
- Comes with several tools:
  - Ontology Visualizer/Editor
  - OntoRootGazetteer
  - Ontology support in JAPE
- Fast in memory repository, scales to millions of statements (depending on RAM)
- Ontology API: JAPE RHS can access Ontology object
  - Useful to ontology population and ontology Learning.

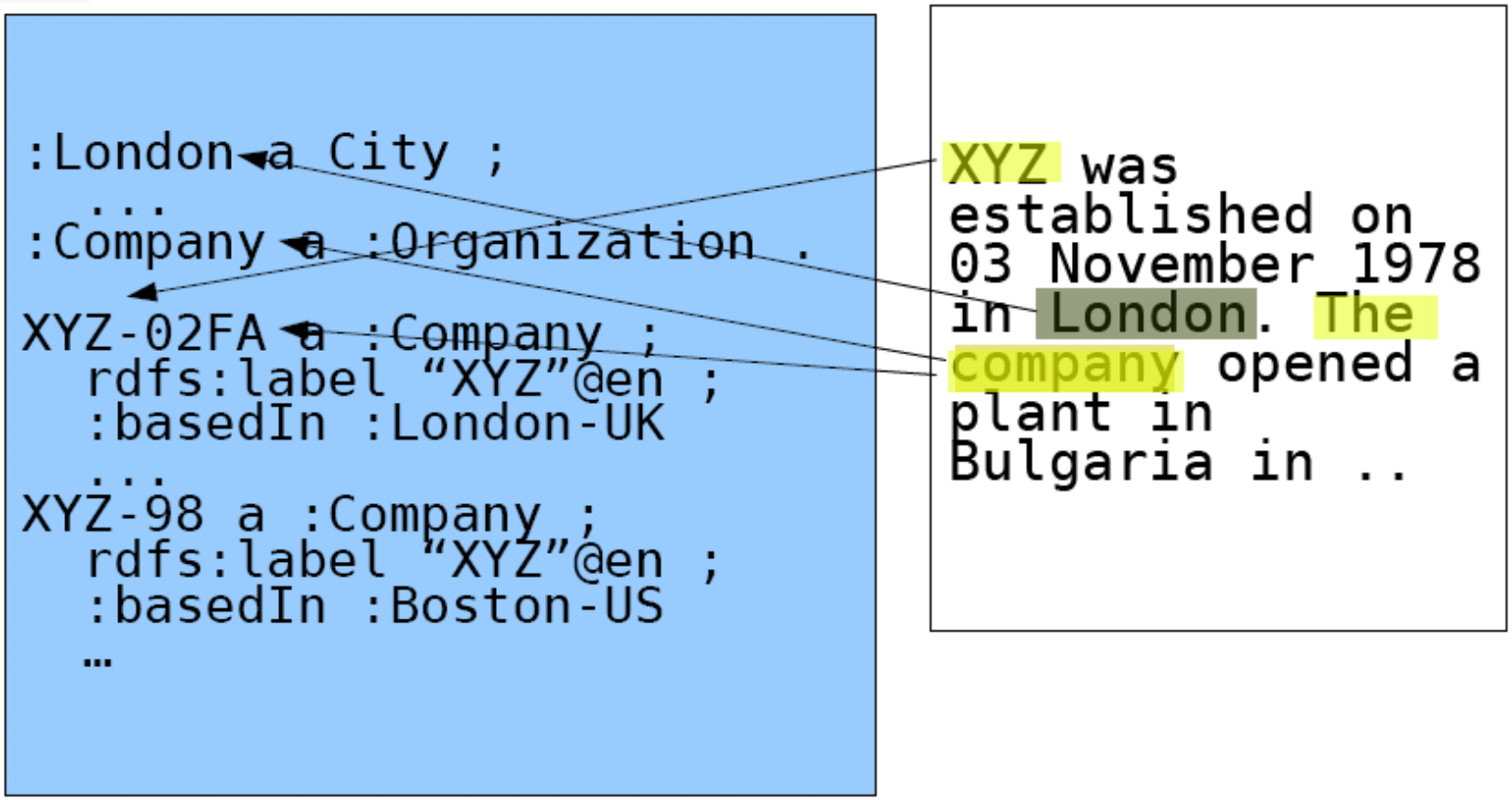
Ontologies

Ontology Editor

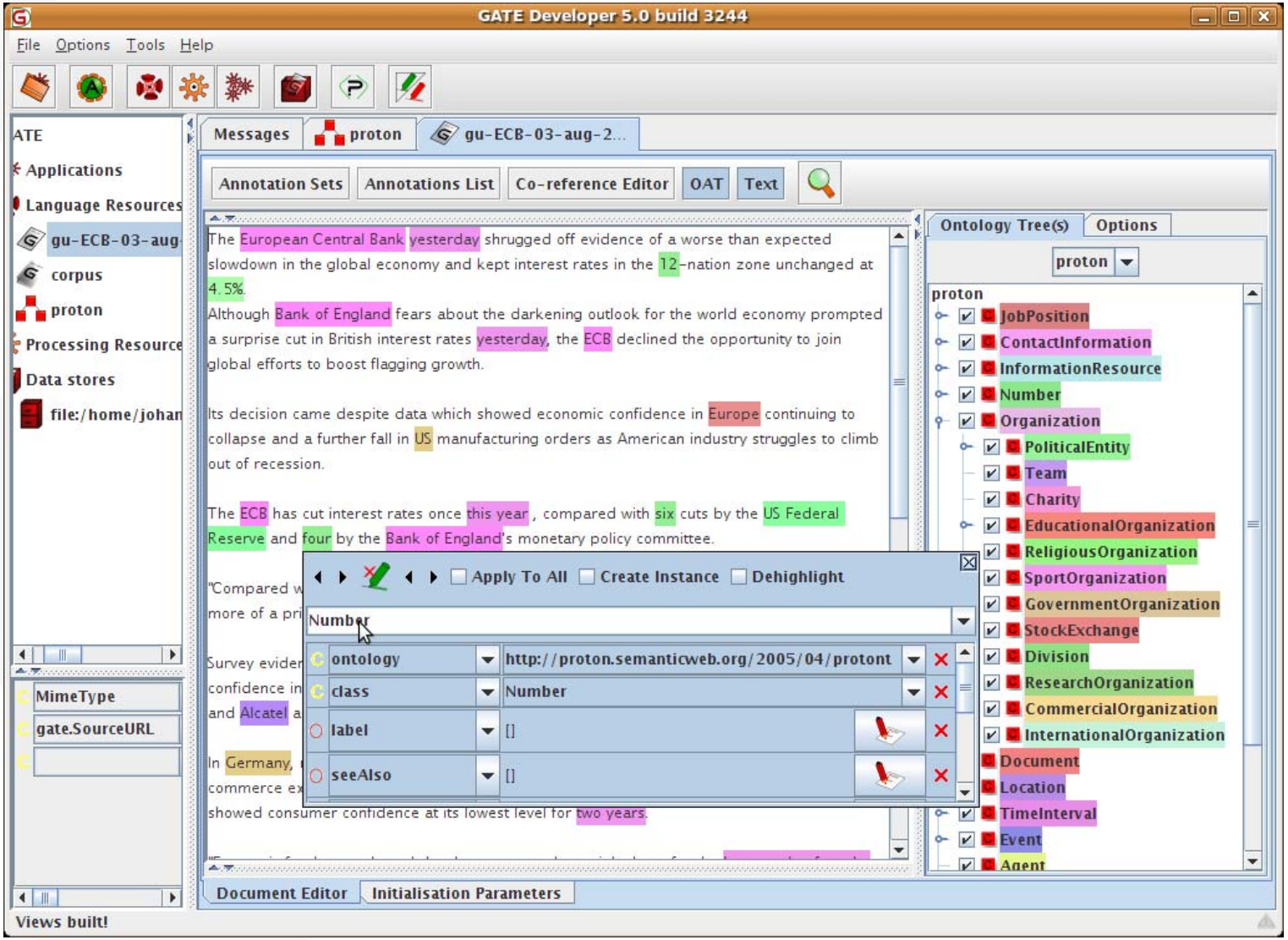


- **A light-weight upper-level ontology:**
  - **250 NE classes;**
  - **100 relations and attributes;**
  - **200.000 entity descriptions;**
  - covers mostly **NE classes, and** ignores general concepts;
  - includes classes representing **lexical resources**




:London a City ;
...
:Company a :Organization .
XYZ-02FA a :Company ;
rdfs:label "XYZ"@en ;
:basedIn :London-UK
...
XYZ-98 a :Company ;
rdfs:label "XYZ"@en ;
:basedIn :Boston-US
…
XYZ was established on 03 November 1978 in London. The company opened a plant in Bulgaria in ..

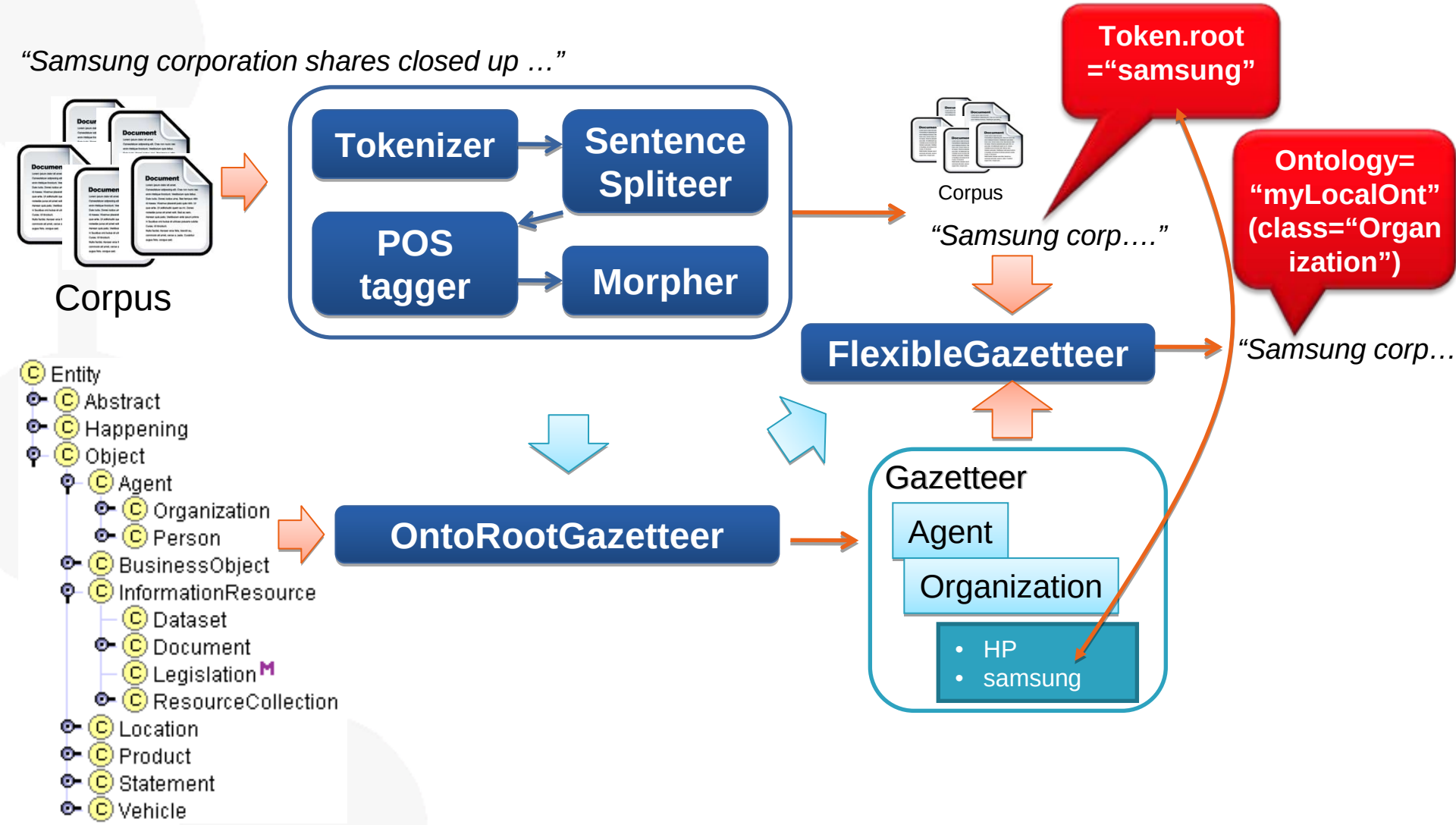
Semantic annotation: Manual



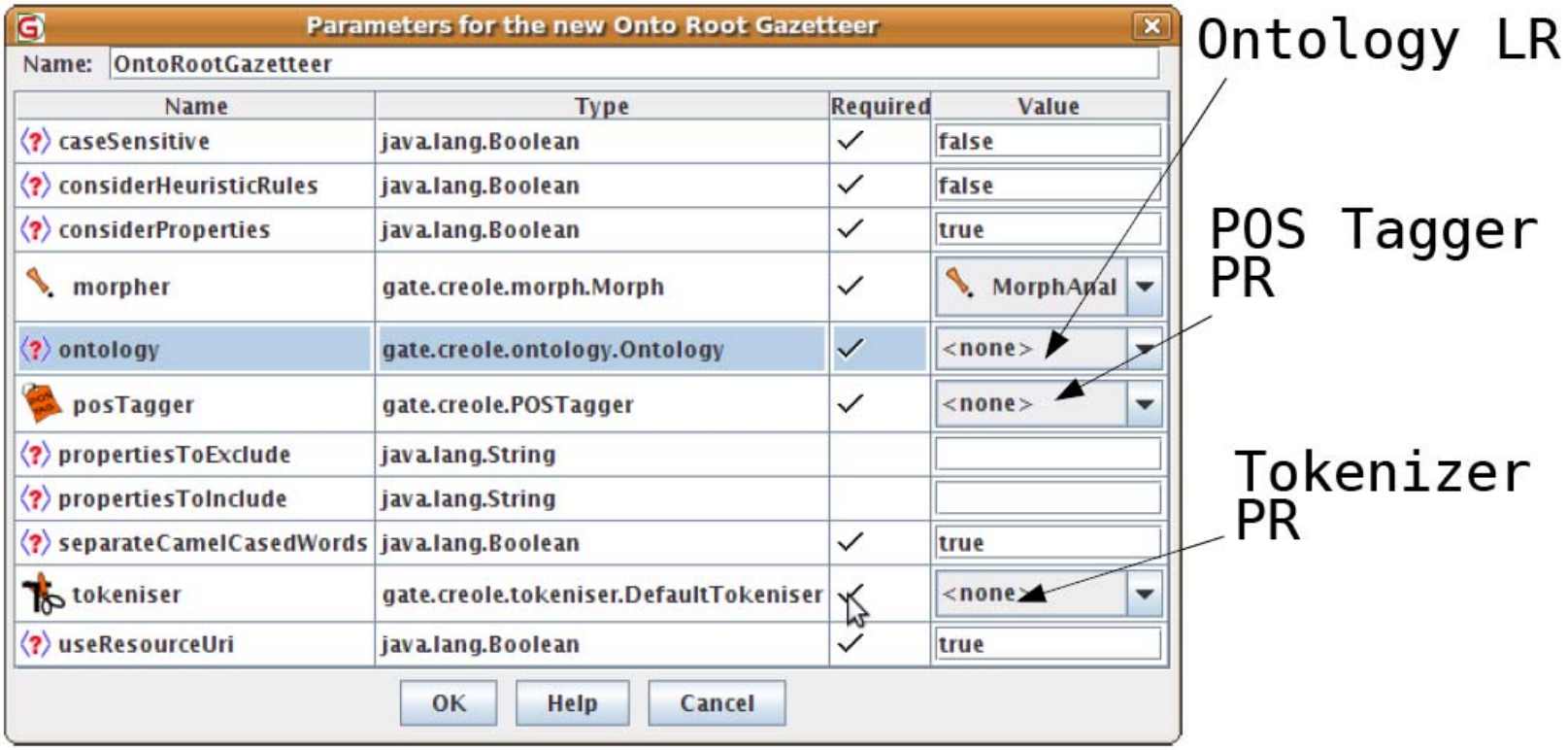
- Onto Root Gazetteer PR
  - Automatic semantic annotation
  - Creates a gazetteer PR that can be used with the FlexibleGazetteerPR

Ontologies

OntoRoot gazetteer



OntoRoot Gazetteer



Jape Rules Method signature:

public void doit(Document doc, Map bindings, AnnotationSet annotations, AnnotationSet inputAS, AnnotationSet outputAS, Ontology **ontology**)
throws JapeException

Ontology API

URI uri = new URI("http://my.uri/#Class1",false);
Oclass **c** = ontology.**addClass**(uri);
Oinstance **i** = ontology.a**ddOInstance**(uri2,c);
Datatype dt = new Datatype(XMLStringURI);
DatatypeProperty **dtp** =
ontology.**addDatatypeProperty**(uri3,domain,dt);
i.**addDatatypePropertyValue**(dtp,new Literal("thevalue"));

Outline

- Introduction to GATE
- ANNIE
  - ANNIE Demo
- Jape Rules
  - Jape Rules Demo
- Machine Learning
- Ontology support
  - Semantic Annotation
  - Ontology Learning
    - **Semantic Annotation Demo**
- LSPs Demo

Manual semantic annotation

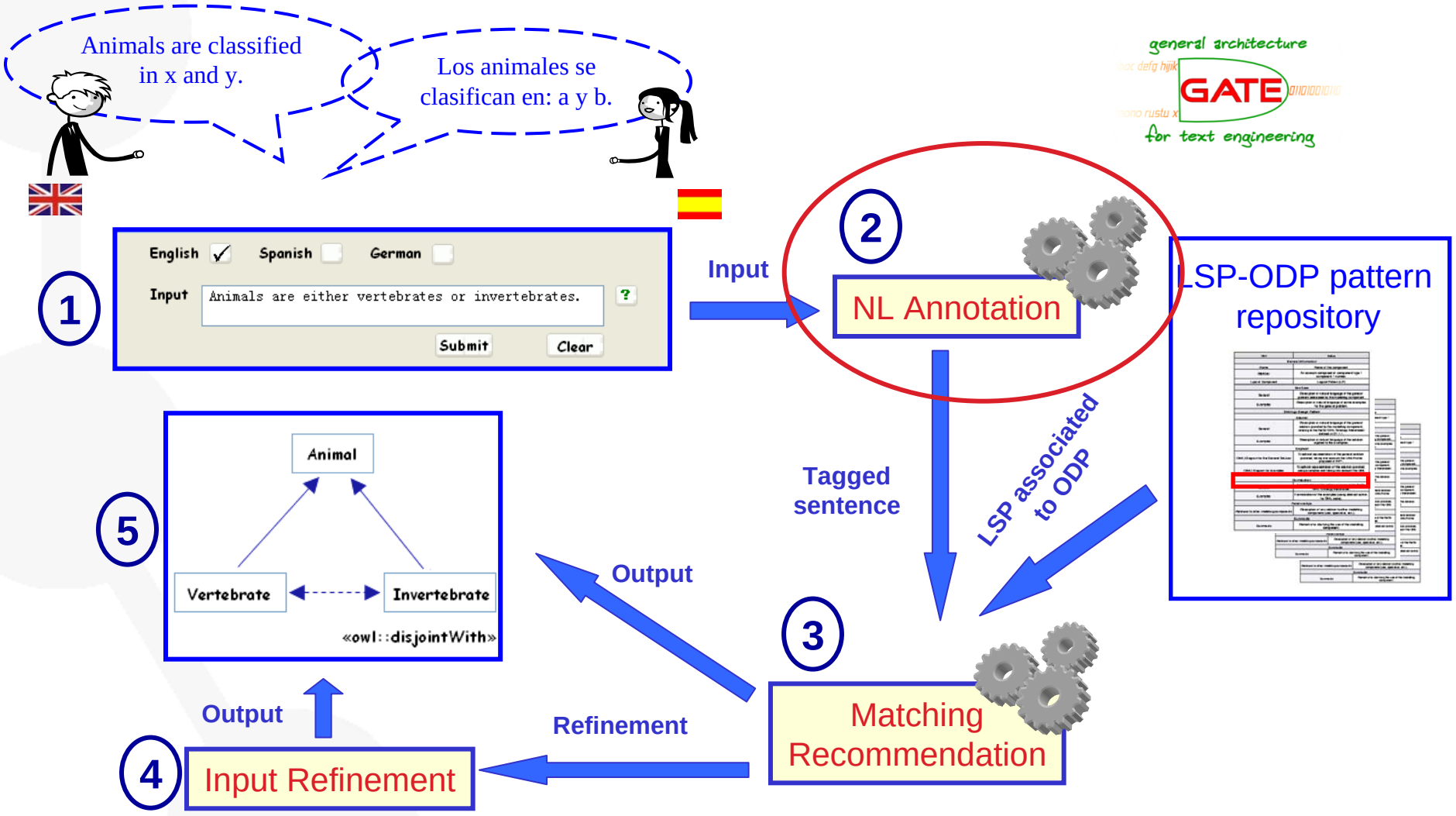
- Load ontology_tools pluging
- Load ontology_OWLIM2
- Language resource OWLIMOntologyLR
- Use OAT in one of the docs in Corpus
- Create annotation

Outline

- Introduction to GATE
- ANNIE
  - ANNIE Demo
- Jape Rules
  - Jape Rules Demo
- Machine Learning
- Ontology support
  - Semantic Annotation
  - Ontology Learning
  - Semantic Annotation Demo
- **LSPs Demo**

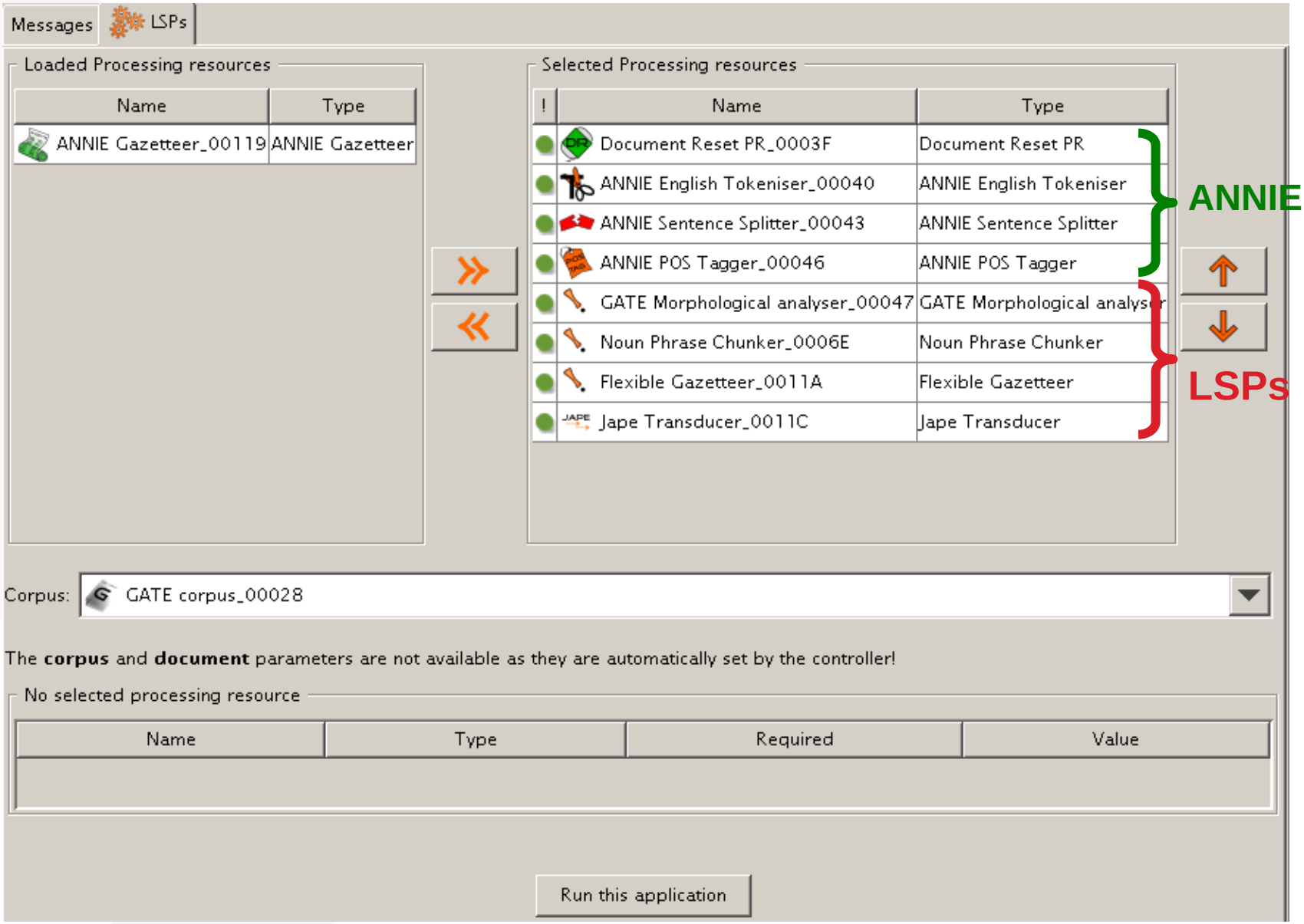
LSPs Demo

S.O.S system



LSPs Demo

PRs



- GATE Morphological Analyser: finds the root and and affixes of a token
  - E.g.: infinitive form of the verb – to be (is, are, was, etc.)
  - singular form of a noun – company (companies)
- Noun Phrase Chunker: identifies nominal groups that have the same function
  - E.g.: Thyroid medicines belong to the general group of hormone medicines
- Flexible Gazetteer: wordlists grouped in different categories to perform NE or Key Phrase Lookups

Introduction to GATE

Andrés García-Silva, Elena Montiel-Ponsoda

Ontology Engineering Group
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo s/n
28660 Boadilla del Monte, Madrid, Spain

http://www.oeg-upm.net

emontiel@fi.upm.es
hagarcia@delicias.dia.fi.upm.es
Phone: 34.91.3363670
Fax: 34.91.3524819

Date: October 22nd, 2009