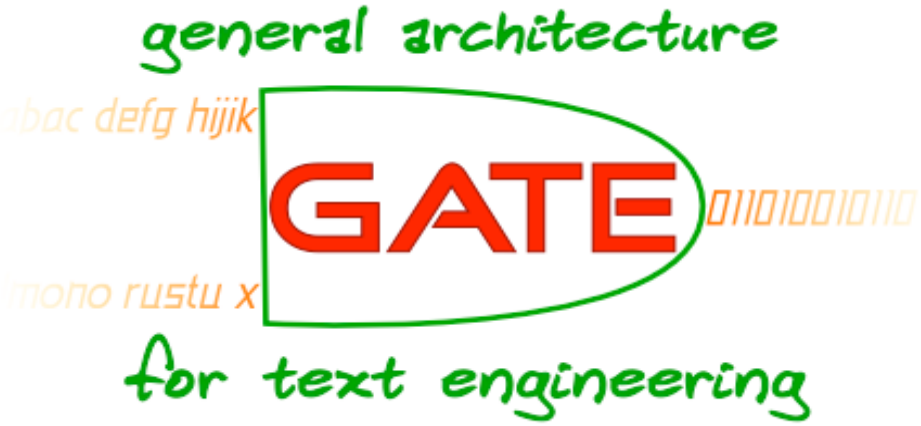


Introduction to GATE Developer

Ian Roberts

Overview

- The GATE component model (CREOLE)
- Documents, annotations and corpora
- Processing components and applications
- Large corpora and data stores

The GATE component model

- CREOLE
  - Collection of RE-usable Objects for Language Engineering
- GATE components: modified Java Beans with XML configuration
- The minimal component = 10 lines of Java, 3 lines of XML, 1 URL
- Why bother?
- Allows the system to load arbitrary language processing components

Types of components

- **Language Resources (LRs)**, e.g. lexicons, corpora, ontologies
- **Processing Resources (PRs)**, e.g. parsers, generators, taggers
- **Visual Resources (VRs)**, i.e. visualisation and editing components
- Resources grouped into *plugins*
- Algorithms are separated from the data, which means:
  - the two can be developed independently by users with different expertise.
  - alternative resources of one type can be used without affecting the other, e.g. a different visual resource can be used with the same language resource

Core LRs - Documents and Corpora

- Central data representation used by GATE
- Document = text + annotations + features
- Corpus = collection of documents

Annotations and Features

- Linguistic information in documents is encoded in the form of annotations
- The annotations associated with each document are a structure central to GATE.
- Each **annotation** consists of
  - start offset
  - end offset
  - a set of features associated with it
  - each feature has a name and a relative value (arbitrary Java object, incl. String)

Annotation sets

- Annotations are grouped in annotation sets
  - e.g. separate sets for gold-standard and machine annotations
- Documents and corpora also have features, which describe them

Annotations Example

Text				
Cyndi savored the soup.				
|0...|5...|10..|15..|20				
Annotations				
Id	Type	SpanStart	Span 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	name_type=person
7	sentence	0	23	

- Similar models
  - TIPSTER
  - ATLAS

I/O Formats in GATE

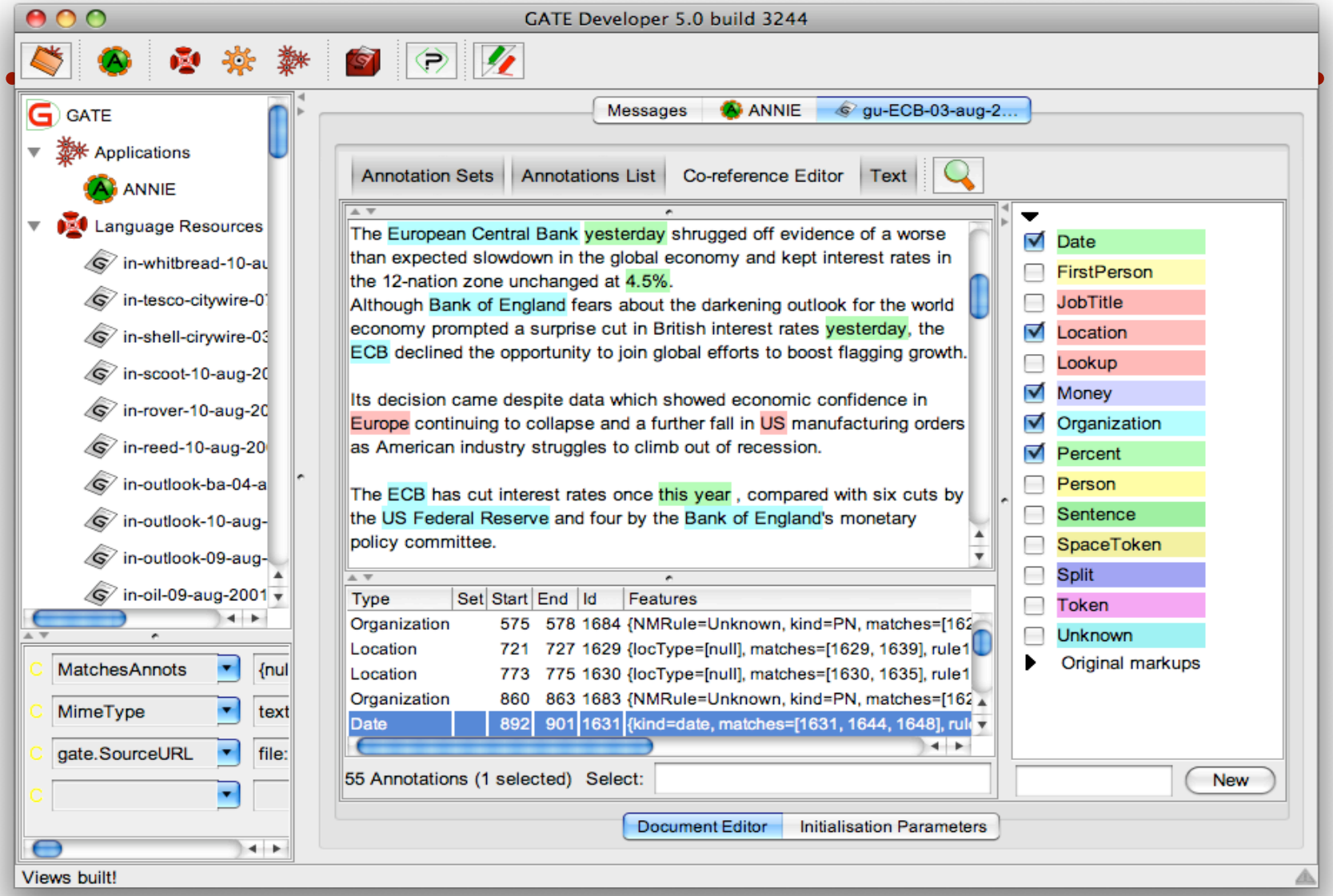
- GATE operates on plain text
- *Document formats* support reading other formats
  - XML, HTML, SGML - tags to annotations
  - Email, plain text - simple paragraph breaks, mail headers, etc.
  - PDF and (some) MS Word - just extract plain text
- Several types of XML dump are available:
  - format-preserving
  - GATE XML persistence format (stand-off), similar to XCES

GATE XML Example

```
<TextWithNodes>
    <Node id="0"/>A TEENAGER
    <Node id="11"/> yesterday <Node id="20"/> accused
    his parents of cruelty by feeding him a daily diet of chips which
    sent his weight ballooning to 22st at the age of l2.
    <Node id="147"/>
</TextWithNodes>


<AnnotationSet>
    <Annotation Type="Date" StartNode="11"  EndNode="20">
      <Feature>
      <Name className="java.lang.String">kind</Name>
      <Value className="java.lang.String">date</Value>
    </Feature>
  </Annotation>
  <Annotation Type="Sentence" StartNode="0"  EndNode="147">
  </Annotation>
</AnnotationSet>
```

The GATE Developer GUI



GUI walkthrough

- Plugins loaded and unloaded using plugin manager (File -> Manage CREOLE plugins)
- When loading HTML/XML documents, tags are converted to annotations in the "Original markups" annotation set.
- Document editor allows editing of the document text - annotations after the edit are repositioned automatically.
- To save a document in GATE XML format, use "Save As Xml…" on the right-click menu

GUI walkthrough (2)

- Documents grouped together into corpora (plural of corpus)
- Three options to create a corpus
  - Create an empty corpus, add loaded documents to it
  - Create an empty corpus and "populate" it by reading files from a directory
  - To create a single-document corpus, right click on the document and select "New corpus with this document"

Hands-on exercise (1)

- Start up GATE Developer
- Load a document
  - Example HTML documents in the `ie\business` directory on USB stick
- Inspect annotations in the "Original markups" set
- Create a corpus and populate it with the example documents

Processing Resources

- Algorithms encapsulated in *Processing Resources* (PRs)
- Simple PRs
  - Document Reset - delete annotations
  - Tokeniser - identify tokens (words, numbers, etc.)
  - Sentence splitter - identify sentence boundaries
- ANNIE (this afternoon)
  - Gazetteer - fast lookup of terms from lists
  - POS tagger - identify nouns, verbs…
  - JAPE finite-state grammars

Processing Resources (2)

- Other PRs include:
  - Co-reference (Tuesday)
  - Machine learning (Wednesday)
  - Ontology tools (Wednesday)
  - Integration of 3rd party tools
    - UIMA (Thursday)
    - Parsers - Minipar, RASP, SUPPLE, Stanford
    - …
- Can take parameters
  - Init parameters
  - Runtime parameters

Applications

- PRs grouped into applications
  - Simple pipeline (run these PRs in this order)
  - Corpus pipeline (run these PRs over each document in this corpus)
- Applications can be saved for future use
- Can be packaged along with their dependencies for deployment on another machine
  - "Export for Teamware"

Hands-on exercise (2)

- Load ANNIE plugin
- Load some PRs
  - Document reset PR
  - English tokeniser (with default parameters)
- Put the PRs into an application
  - Create a corpus pipeline, add the reset PR followed by the tokeniser
  - Run it over your corpus, inspect the results in the document viewer
  - Change a runtime parameter - set tokeniser annotationSetName to another value, run the application again
  - This time the annotations are in your named annotation set
- Save and restore
  - Save the application to a file, Remove the application from GATE and reload from the saved file.

Persistence

- GATE provides *data store* abstraction for persistent storage of LRs
- Useful for processing large corpora
  - When processing a persistent corpus, controller loads documents one by one rather than all at once

Data Store walkthrough

- Several types of data store - most commonly used is "serial data store"
- To create, select an empty directory
- Create empty corpus, save to the datastore
  - Corpus is now considered "persistent"
- When populating a persistent corpus, each document is loaded from disk, saved to the datastore and unloaded from memory before processing the next one
  - Particularly useful for very large corpora

Hands-on exercise (3)

- Create a new SerialDataStore
- Create an empty corpus
- Save it to the datastore
- Populate the corpus as before
- Run your tokeniser application over this corpus, and look at the results