General Architecture for Text Engineering
GATE

Pablo Calleja Ibáñez

Index

- Basic concepts
- Processing Resources
- Gazetteers
- JAPE Rules
- Output
- GATE Embedded
- GATE Architecture
- Creating your own PR

Basic concepts

GATE is:

- An architecture for NLP
- A framework
- An Integrated Development Environment (IDE)

Basic concepts

Three different types of resources:

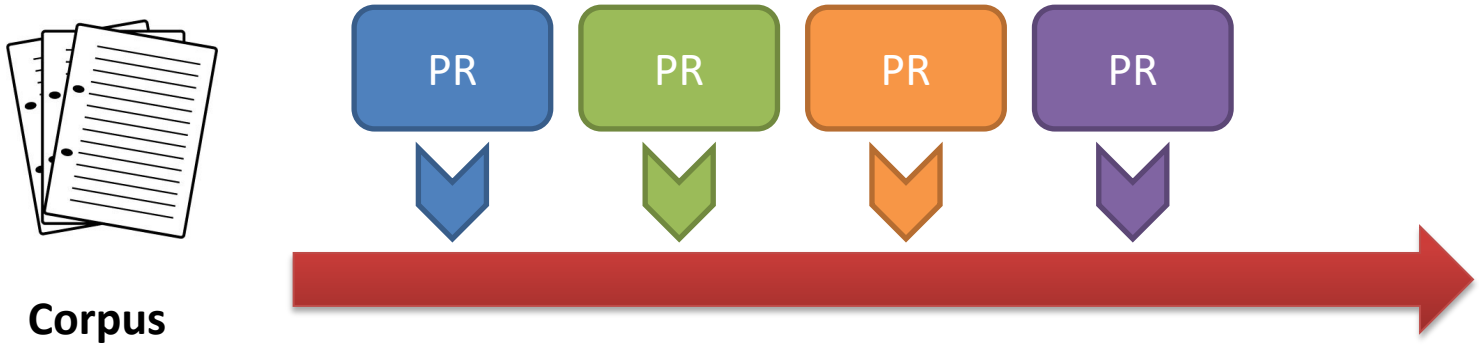
- Language Resources
  - Document
  - Corpus
- Datastores

- Processing Resources

Basic concepts. How it works?

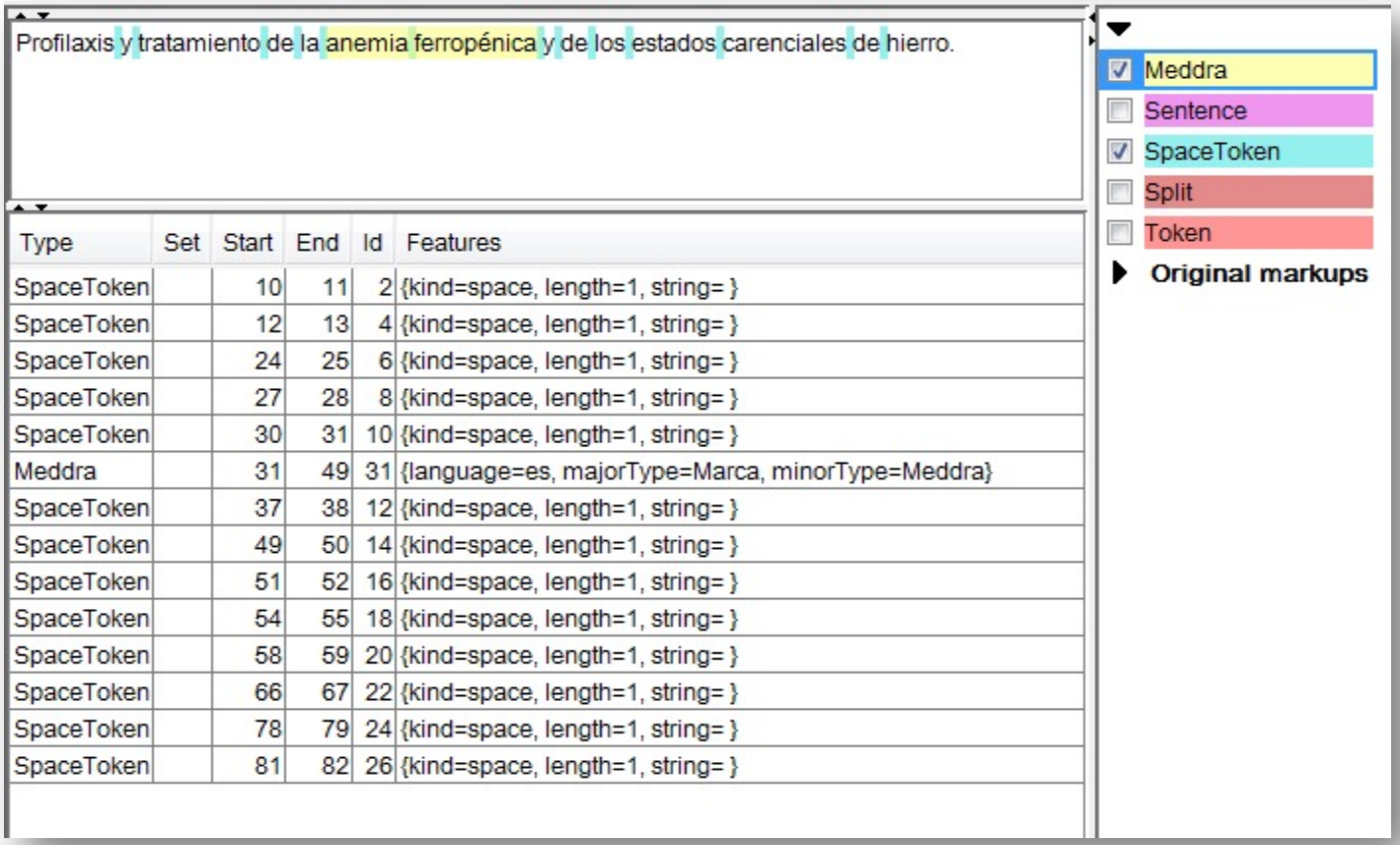
Applications:

- Applications are built by ordered Processing Resources. Each PR applies its functionality and result, as an assembly line.
- Are applied to a corpus. Each document goes through all Processing Resources.



Basic concepts. Annotations

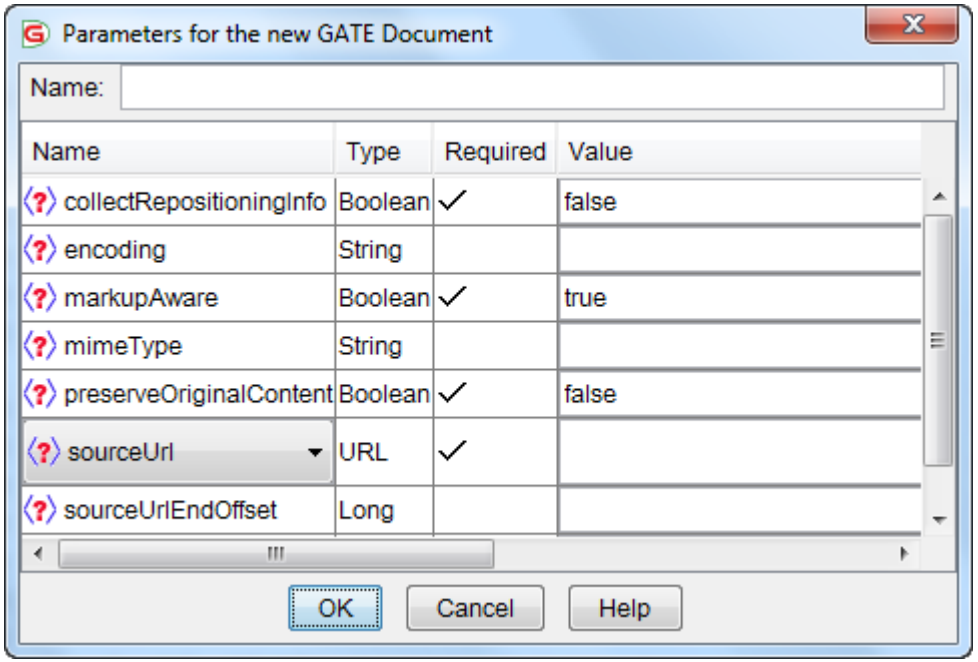
Every product or result over the document in GATE is tagged by annotations.



Type	Set	Start	End	Id	Features
SpaceToken		10	11	2	{kind=space, length=1, string= }
SpaceToken		12	13	4	{kind=space, length=1, string= }
SpaceToken		24	25	6	{kind=space, length=1, string= }
SpaceToken		27	28	8	{kind=space, length=1, string= }
SpaceToken		30	31	10	{kind=space, length=1, string= }
Meddra		31	49	31	{language=es, majorType=Marca, minorType=Meddra}
SpaceToken		37	38	12	{kind=space, length=1, string= }
SpaceToken		49	50	14	{kind=space, length=1, string= }
SpaceToken		51	52	16	{kind=space, length=1, string= }
SpaceToken		54	55	18	{kind=space, length=1, string= }
SpaceToken		58	59	20	{kind=space, length=1, string= }
SpaceToken		66	67	22	{kind=space, length=1, string= }
SpaceToken		78	79	24	{kind=space, length=1, string= }
SpaceToken		81	82	26	{kind=space, length=1, string= }

Basic concepts. Resource features

- Language Resources and Processing Resources have Features



- Only Processing Resources distinguishes initial features and runtime features.

The three basic PR

Most of Information Extraction systems have three main task. GATE incorporates them as PR.

Gazetteers

The role of the gazetteer is to identify entity names in the text based on lists.

de urgencia
de moderada a grave
de moderado a grave
de moderado a intenso
de moderados a graves
de moderada a severa
de inicio parcial
de intensidad moderada
de leve a moderado
de leve a moderada
de leves a moderadas
de leves a moderados

Moderadores.lst

Moderadores.lst:
Moderador:
ModeradoresMedicamentos:
es:
Lookup

Moderadores.def

List Name

Annotation Name

Types

JAPE Rules

- JAPE is a Java Annotation Patterns Engine.
- JAPE allows you to recognise regular expressions in annotations on documents.

```
Rule InitialsRule
({Token.orth== “allCaps"}):init
-->
:init.Initials= {kind = “Initial Label"}
```

JAPE Rules. Syntax

```
Phase: PatronItis
Input: Token
Options: control = first
Rule: ItisRule
(
        (
        {Token.string =~ "itis$"}
        ({Token.kind==word})*
        ):enf
        ({Token.kind== punctuation})
)
-->
:enf.Enfermedad =
{kind =ItisPattern",
string = :enf@cleanString}
```

Phase

Input annotations

Matching style

Rule Name

Rule Label

New Annotation

JAPE Rules. Control styles

- Appelt
- Brill
- All
- First
- Once

```
Phase: ControlStyles
Input: Token
Options: control = ???

Rule: ControlStylesRule
(
(({Token.orth==“upperInitial”})+ ):name
)
-->
:name.Name = {kind = “NameDetected"}
```

Text

Paul **Jimmy** **Brian**

JAPE Rules. Control styles

- Appelt = longest match

```
Phase: ControlStyles
Input: Token
Options: control = Appelt

Rule: ControlStylesRule
(
(({Token.orth==“upperInitial”})+ ):name
)
-->
:name.Name = {kind = “NameDetected"}
```

Text

JAPE Rules. Control styles

- Brill =

every combination

from start of match

```
Phase: ControlStyles
Input: Token
Options: control = Brill

Rule: ControlStylesRule
(
(({Token.orth==“upperInitial”})+ ):name
)
-->
:name.Name = {kind = “NameDetected"}
```

Text

JAPE Rules. Control styles

- All= every combination

```
Phase: ControlStyles
Input: Token
Options: control = All

Rule: ControlStylesRule
(
(({Token.orth==“upperInitial”})+ ):name
)
-->
:name.Name = {kind = “NameDetected"}
```

Text

JAPE Rules. Control styles

- First=

first match

```
Phase: ControlStyles
Input: Token
Options: control = First

Rule: ControlStylesRule
(
(({Token.orth==“upperInitial”})+ ):name
)
-->
:name.Name = {kind = “NameDetected"}
```

Text

JAPE Rules. Control styles

- Once=

exit after first match

```
Phase: ControlStyles
Input: Token
Options: control = Once

Rule: ControlStylesRule
(
(({Token.orth==“upperInitial”})+ ):name
)
-->
:name.Name = {kind = “NameDetected"}
```

Text

Paul	**Jimmy**	**Brian**

JAPE Rules. Input and combinations

```
Phase: PatronIndicadoEn
Input: Token Lookup
Options: control = first
Rule: IndicadoRule
(

{Token.string == "indicado"}
{Token.string == "para"}
({Token.string == "el"} |  {Token.string == "la"})
({Token.kind==word,Token.string!=“mejora”})+:enf
({Lookup} | {Token.kind==punctuation})
)
-->
:enf.Enfermedad = {kind = "IndicadoPattern",string = :enf@cleanString}
```

JAPE Rules. Using JAVA

```
Rule: ToLowerCase
({Token}):word
-->
{
  AnnotationSet set = bindings.get(“word");
  Annotation annotation = set.iterator().next();
  FeatureMap fmap= annotation.getFeatures();

  String newFeature= (String) fmap.get(“string");


  fmap.put(“minus", newFeature.toLowerCase());
}
```

JAPE Rules. Negation

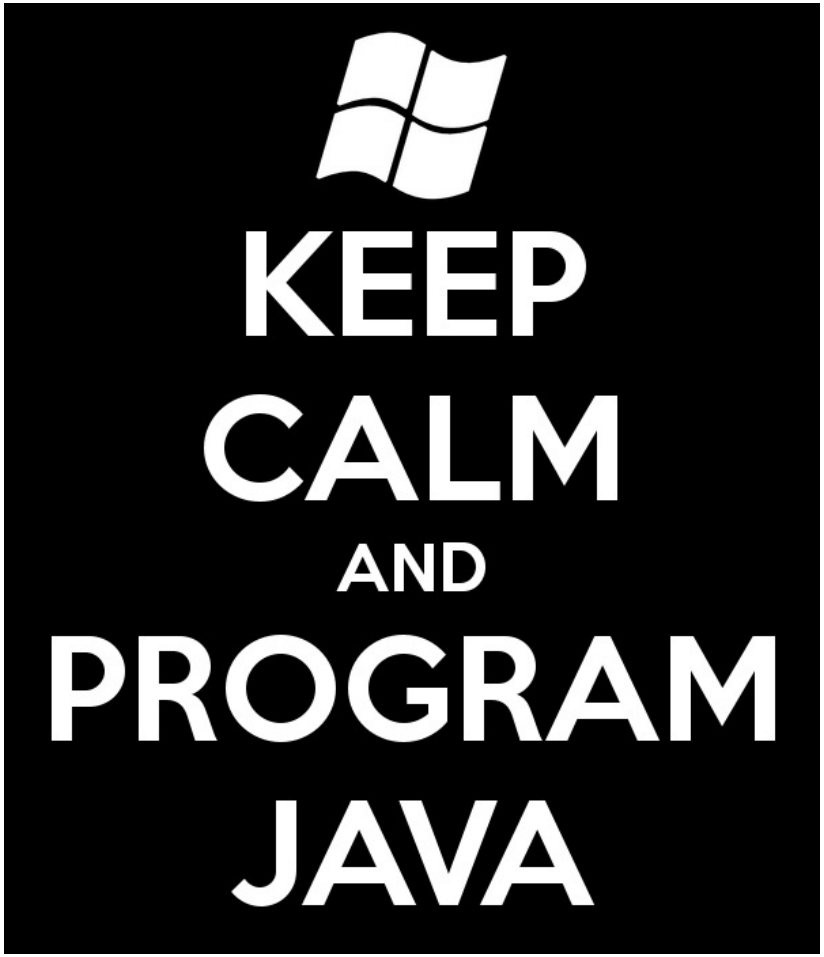
- JAPE also supports negative constraints
- However, there is no support for a general negative operator to prevent a particular sequence of annotations to be found.
- Other techniques are applied.

Output Format

How GATE saves the information of the document?

- In a XML file with all the annotations and features of the document.
- That makes the documents hard to reuse outside GATE.

Do it yourself!

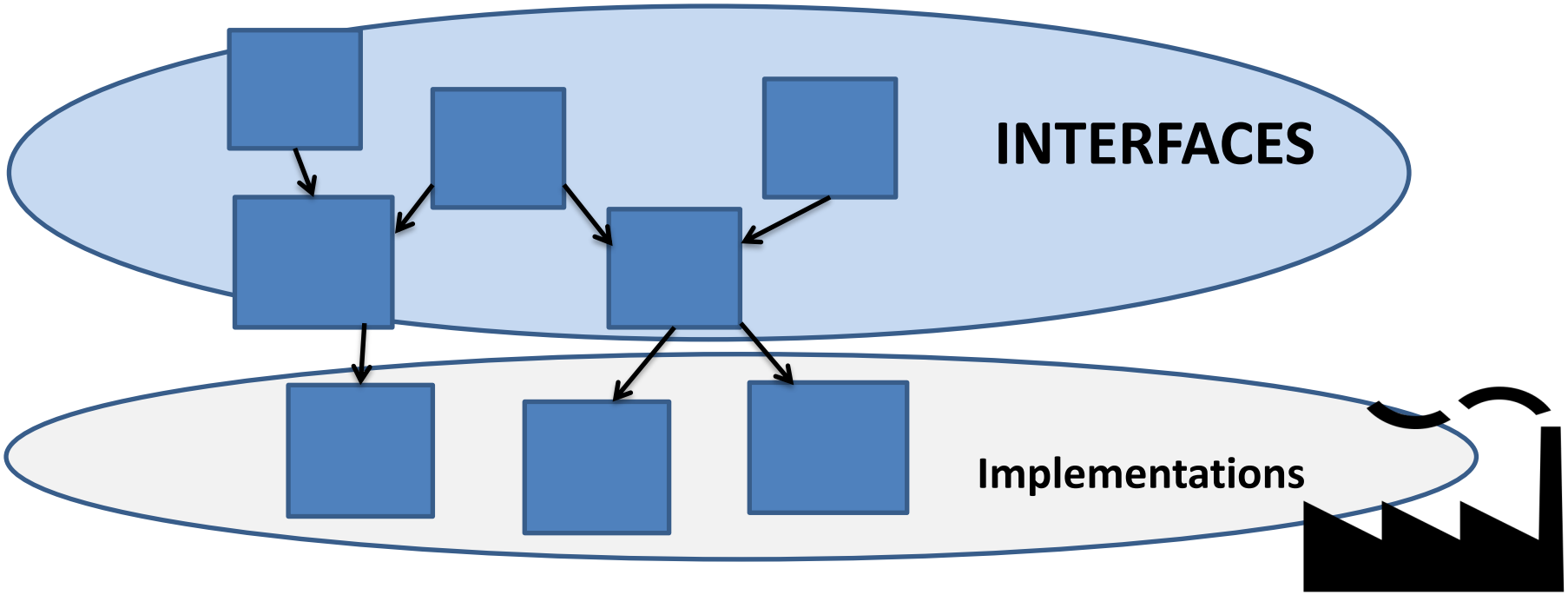


GATE Embedded

- The graphic IDE consumes machine resources and waste a lot of processing time with big amount of data.
- Also there is no support for automatic task.
- It is possible to make more complex jobs and adapted to you.

GATE Architecture

GATE has a huge number of classes. However, most of them are interfaces and the implementations are restricted to a small group of classes which are created by a factory.



Creation of a PR

- Three main methods:
  - Init()
  - Execute()
  - Reinit()
- Features:
  - Init features
  - Runtime features

Compile the java class and register the .jar in a XML

Load the plugin

To load a new Collection of Reusable of Language Engineering (CREOLE), select the folder where the jar and the XML are.