Text Analytics Tutorial

The 7th Computer Science and Electronic Engineering Conference (CEEC) 2015

Lab

Named Entity Recognition

Maha Althobaiti, Udo Kruschwitz, Massimo Poesio

September 23, 2015

Definition

- Cornerstone of IE
- Identification of proper names in texts,
- Classification them into a set of predefined categories
  - Person
  - Organization (companies, government organizations, committees, etc.)
  - Location (cities, countries, rivers, etc.)
  - Date and time expressions
- Other types are frequently added, as appropriate to the Application.
  - Medical domain
  - Biological domain

Example of NER

Steven Paul Jobs (February 24, 1955 – October 5, 2011) was an American businessman. He was best known as the co-founder, former chairman of Apple Inc.

Apple Inc.'s world corporate headquarters are located in the middle of Silicon Valley.

Entities:

- Steven Paul Jobs → Person
- Apple Inc. → Organisation
- Silicon Valley → Location

NER Classifiers

- Using ready-made NE classifiers
  - Stanford NE recognizer
  - OpenNLP NE recognizer
  - GATE
- Building specialised NE classifiers
  - CRF++

Ready-made NE Classifiers

Ready-made NE classifiers

- There are many ready-made NE classifiers that can recognize pre-specified set of NE types. They are also pre-trained on certain domains.

- Examples:
  - Stanford NE recognizer
  - OpenNLP NE recognizer
  - GATE

OpenNLP Name Finder

- Written in Java.
- Can be used:
  - As stand-alone tools.
  - As plugins in other frameworks.
- Based on maximum entropy to recognize types of different entities: Persons, Locations, organizations, dates, times, money, and percentages.
- Has a set of Ready-made models trained on various freely available corpora

Practical Work

- OpenNLP name finder can be tested on a raw text by using a command line tool as follows:
  - Download apache-opennlp-1.5.3-bin.zip file from http://mirror.catn.com/pub/apache/opennlp/
  - Unzip the file to your desktop
  - Download the English person model en-ner-person.bin from http://opennlp.sourceforge.net/models-1.5/
  - store the model in the distribution directory
  - Using command prompt, change the current directory to apache-opennlp-1.5.3-bin directory, then type:

```
java -jar lib\opennlp-tools-1.5.3.jar TokenNameFinder en-ner-person.bin
```

  - The name finder now is ready to read from standard input, copy and paste a raw text to *command prompt* or just type a sentence using keyboard.
  - The name finder will output the text with markup for person names.

Note: ‘en-ner-person.bin’ is a person model to detect only person names from text.

OpenNLP Location Finder

- Input

Sochi is a city in Krasnodar Krai , Russia , located on the Black Sea coast near the border between Georgia and Russia .

- Output

Sochi is a city in Krasnodar Krai , **<START:location>** Russia **<END>** , located on the **<START:location>** Black Sea **<END>** coast near the border between **<START:location>** Georgia **<END>** and **<START:location>** Russia **<END>** .

```
C:\apache-opennlp-1.5.3>java -jar lib\opennlp-tools-*.jar TokenNameFinder en-ner
-location.bin
Loading Token Name Finder model ... done (0.800s)
Sochi is a city in Krasnodar Krai , Russia , located on the Black Sea coast near
 the border between Georgia and Russia .
Sochi is a city in Krasnodar Krai , <START:location> Russia <END> , located on t
he <START:location> Black Sea <END> coast near the border between <START:locatio
n> Georgia <END> and <START:location> Russia <END> .
```

Try…

- Using a command line tool, test OpenNLP name finders for other NE types (e.g., money, date, time, organization)
  - All ready-made models for different NE types can be found at http://opennlp.sourceforge.net/models-1.5/

OpenNLP Name Finder API

In order to embed OpenNLP name finder into your application, two main steps should be conducted:

1. The Model must be created and loaded into memory as shown below:

```
InputStream modelIn = new FileInputStream("en-ner-person.bin");

  TokenNameFinderModel model = new TokenNameFinderModel(modelIn);
  if (modelIn != null) {
      modelIn.close();
  }
```

2. After the model is loaded the NameFinderME can be instantiated.

```
NameFinderME nameFinder = new NameFinderME(model);
```

Note: do not forget to include necessary jar files into the classpath opennlp-maxent-3.0.3.jar, opennlp-tools-1.5.3.jar

OpenNLP Name Finder API - Example

The following sample code shows an example of using OpenNLP NER API.

```
public static void main(String[] args) throws InvalidFormatException, IOException
{
ArrayList nameslist = new ArrayList();

String text="Text to be processed";

// tokenize the text before detecting NEs from the text
InputStream modelTok = new FileInputStream("en-token.bin");
TokenizerModel modelTokenizer = new TokenizerModel(modelTok);
  if (modelTok != null)
  {
    try {
    modelTok.close();
        }
    catch (IOException e) {}
  }
//create an instance of learnable tokenizer and initialise it with model
Tokenizer tokenizer = new TokenizerME(modelTokenizer);
String tokens[] = tokenizer.tokenize(text);
```

OpenNLP Name Finder API - Example

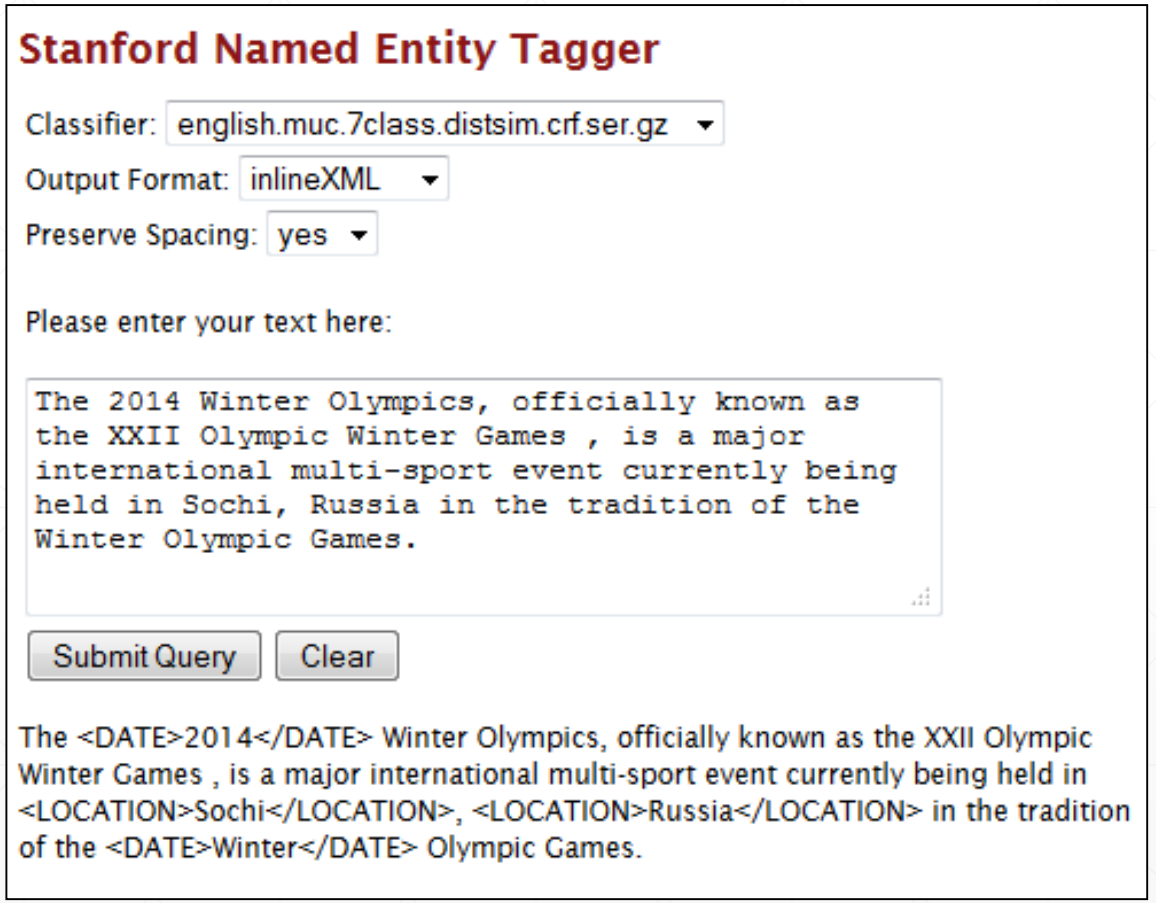
```
// load the model
  InputStream modelIn = new FileInputStream("en-ner-person.bin");
  TokenNameFinderModel model = new TokenNameFinderModel(modelIn);
  if (modelIn != null)
  {
      modelIn.close();
  }
// instantiate the NameFinder
  NameFinderME nameFinder = new NameFinderME(model);
  Span nameSpans[] = nameFinder.find(tokens);
    for (int i=0;i<nameSpans.length;i++)
        {
    for (int j=nameSpans[i].getStart();j<nameSpans[i].getEnd();j++)
                {
        nameslist.add(tokens[j]);
                }
        }

    for (int k=0;k<nameslist.size();k++)
    System.out.println(nameslist.get(k));
 }
```

Stanford NE Recognizer (CRF Classifier)

- Written in Java
- Based on Conditional Random Field sequence model
- For English, Chinese, and German
- 3 types of Models:
  - 3 class Model: Location, Person, Organization
  - 4 class Model: Location, Person, Organization, Misc
  - 7 class Model: Time, Location, Organization, Person, Money, Percentage, Date
- interfaces to Stanford NER
  - UIMA
  - Text-NLP-Stanford-EntityExtract (Perl)
  - ruby-nlp (Ruby)
  - Pyner (Python)

Online Demo

http://nlp.stanford.edu:8080/ner/



Stanford NE Recognizer

MODELS / TRAINING DATA

Three different models are included with Stanford NER package:

1. **english.all.3class.distsim.crf.ser.gz:** a 3 class NER tagger that can label Person, Organization, and Location entities. It is trained on data from CoNLL, MUC6, MUC7, and ACE.
2. **english.conll.4class.caseless.distsim.crf.ser.gz:** a 4 class NER tagger that can label Person, Organization, Location, and Misc. It is trained on the CoNLL 2003 Shared Task.
3. **english.muc.7class.caseless.distsim.crf.ser.gz:** It is trained only on data from MUC and distinguishes between 7 different classes

Stanford NE Recognizer API

- Stanford NER API offers different ways to easily control the format of the output and to directly embed NE recognizer into your application instead of using command line interface.
- For more information on API calls, look in the javadoc included in the distribution directory, look first at the edu.stanford.nlp.ie.crf package and CRFClassifier class.
- Online javadoc:
  http://nlp.stanford.edu/nlp/javadoc/javanlp/

Named Entity Recognition with GATE

- GATE is distributed with an IE system called ANNIE.
- ANNIE relies on finite state algorithms and the JAPE language.
- ANNIE components form the following pipeline:
  - Tokeniser
  - Sentence Splitter
  - POS tagger
  - Gazetteers
  - Semantic tagger (JAPE transducer)
  - Orthomatcher (orthographic coreference)
- You can try ANNIE tool online at:

  http://services.gate.ac.uk/annie/

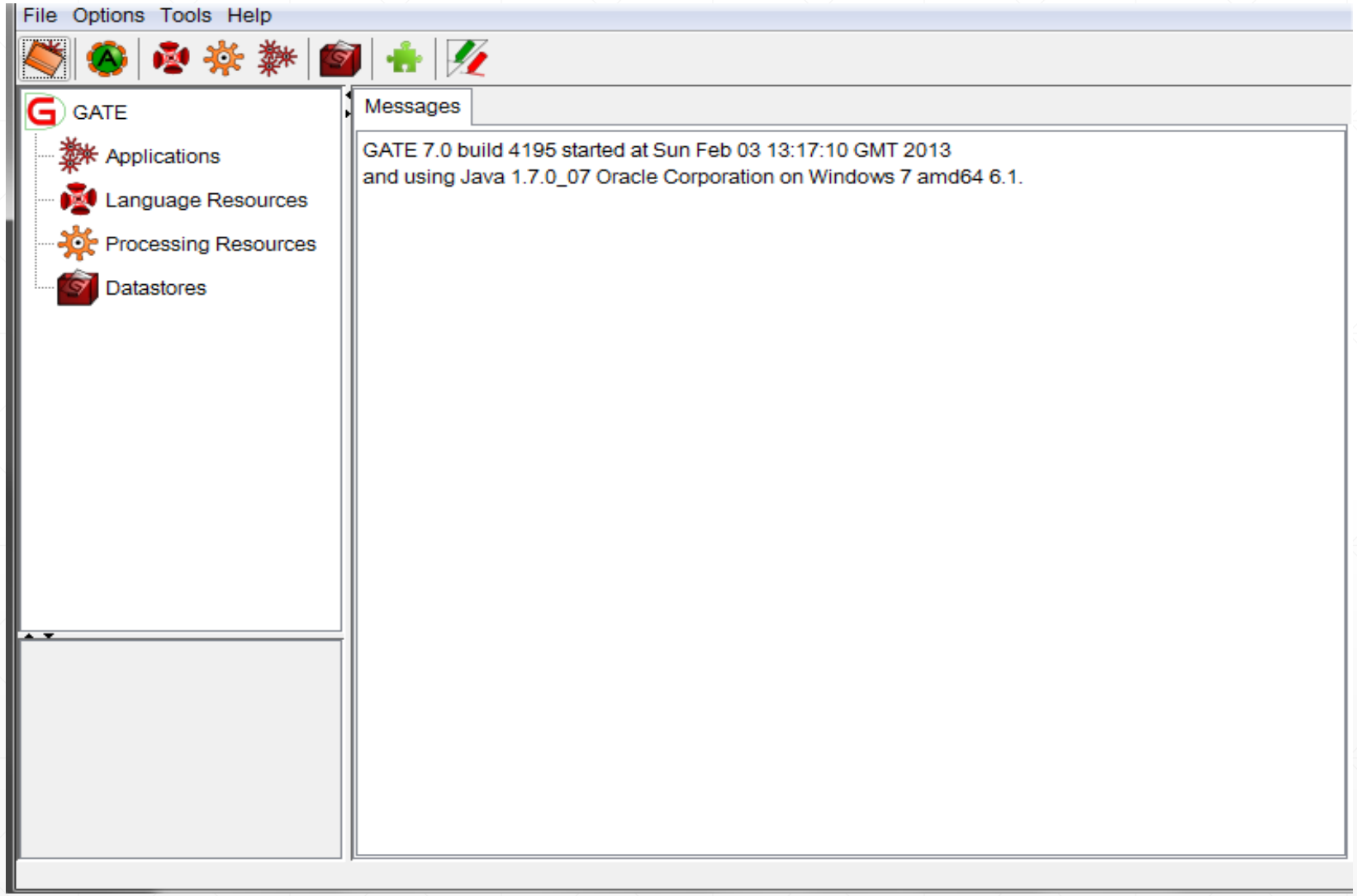
Launch GATE

- Start → All Programs → GATE developer 8.0

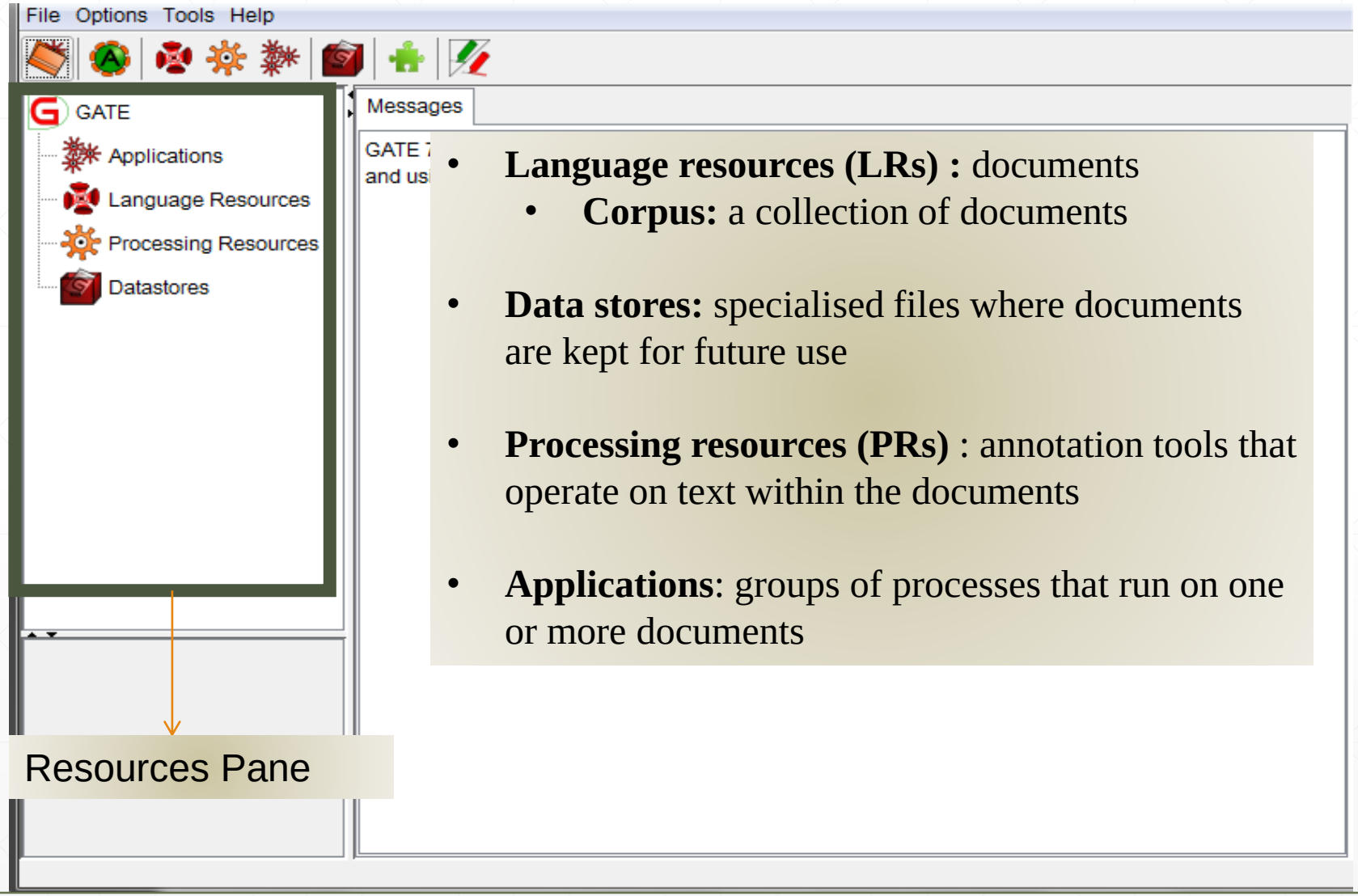
  OR

- Run the batch file (gate.bat) exist in your GATE home directory

GATE Main Window

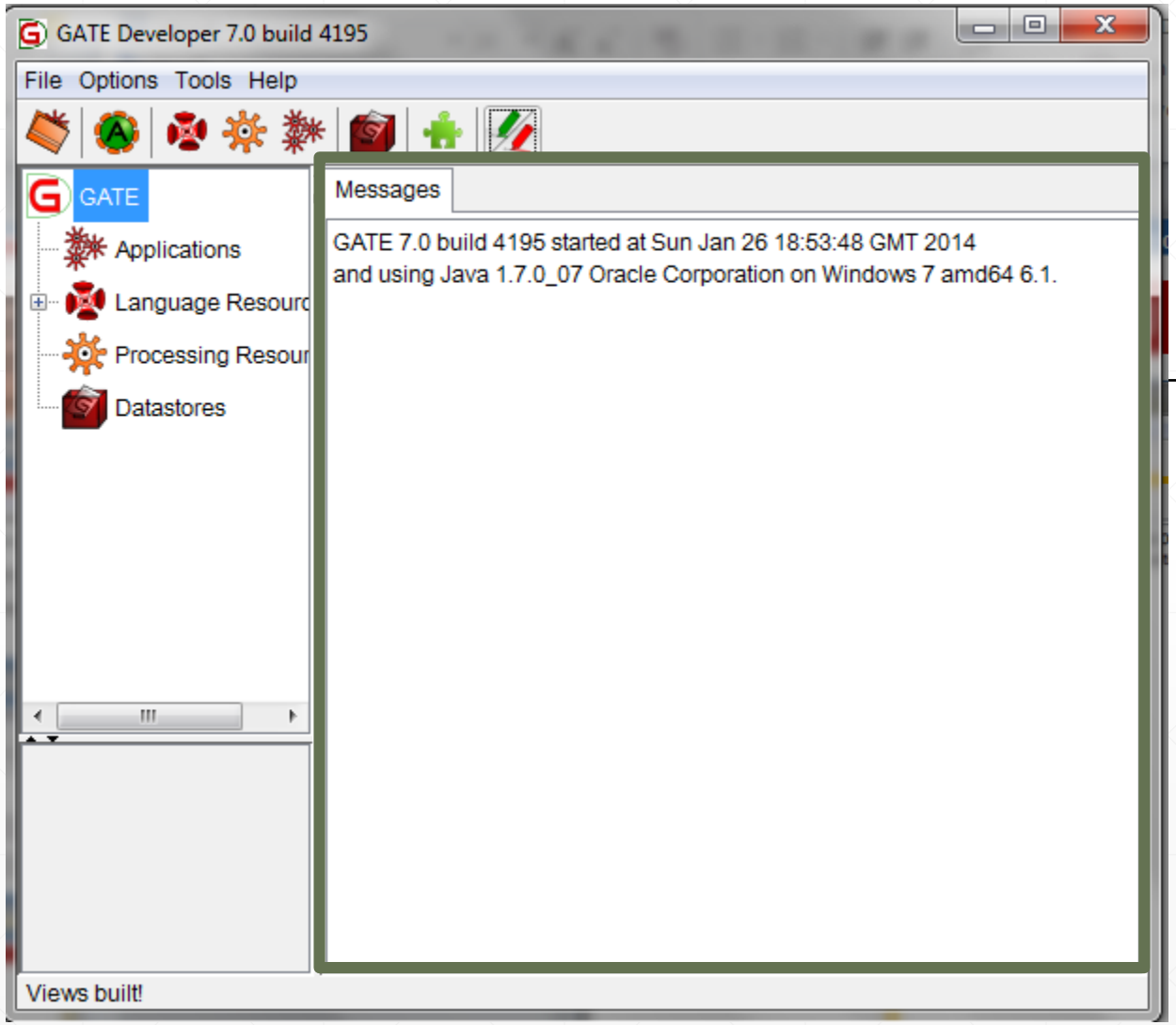


Guided Tour to GATE GUI



- **Language resources (LRs) :** documents
  - **Corpus:** a collection of documents
- **Data stores:** specialised files where documents are kept for future use
- **Processing resources (PRs)** : annotation tools that operate on text within the documents
- **Applications**: groups of processes that run on one or more documents

Guided Tour to GATE GUI

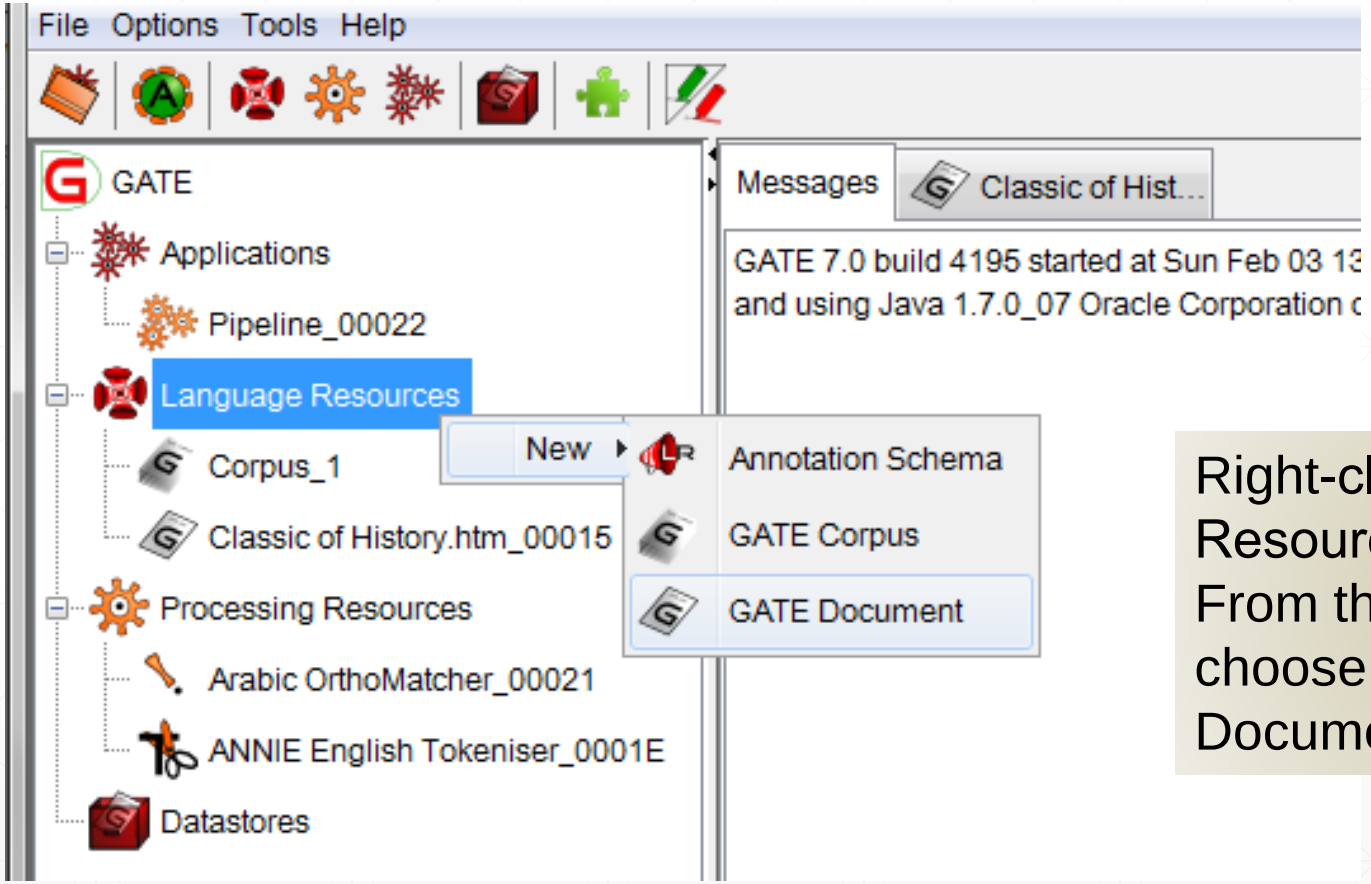


Display Pane: contains the current element you are working with. In case you open GATE for the first time, it only shows a system message.

Loading Document in GATE

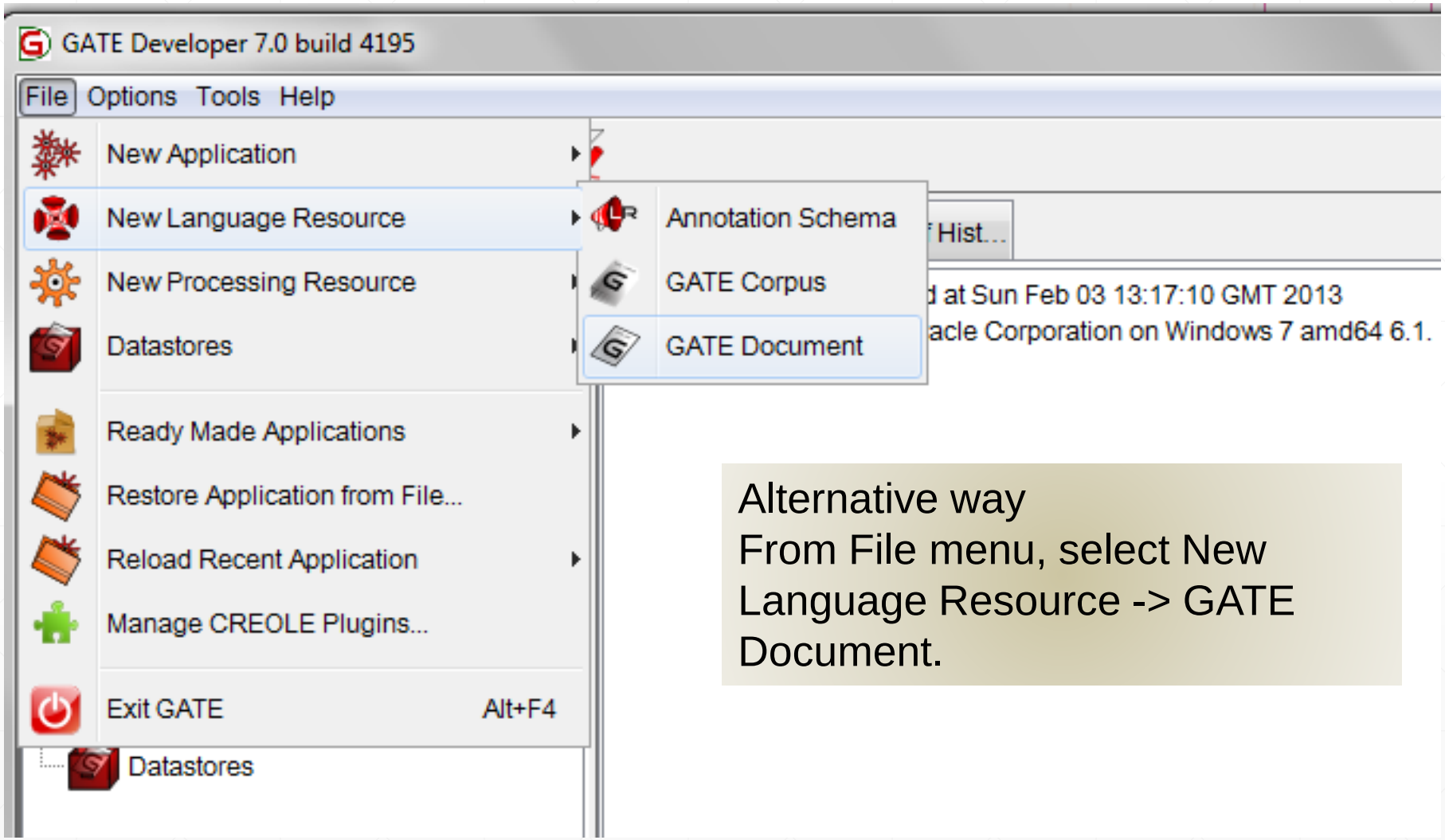
- GATE can process documents in all kinds of formats: plain text, HTML, XML, PDF, Word etc.
- When GATE loads a document, it converts it into a special format for processing.
- Documents can be exported in various formats or saved in a datastore for future processing within GATE.

Loading Document in GATE



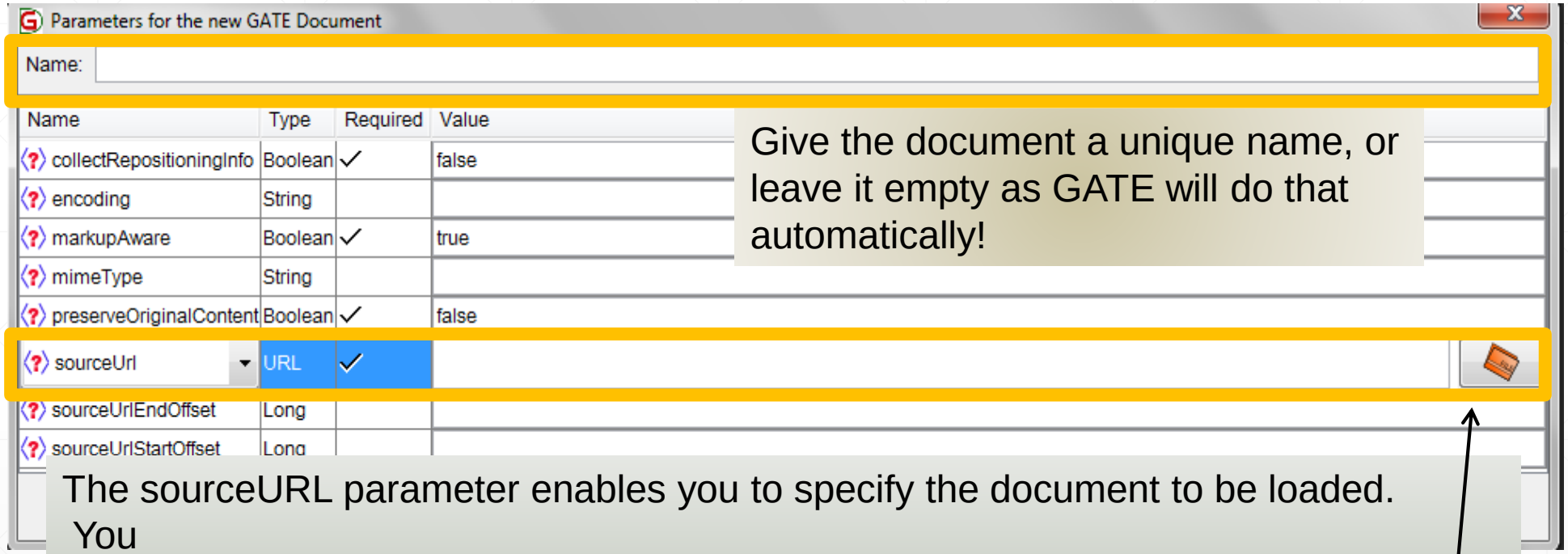
Right-click on ‘Language Resources’.
From the appearing menu, choose New->GATE Document.

Loading Document in GATE



Alternative way
From File menu, select New Language Resource -> GATE Document.

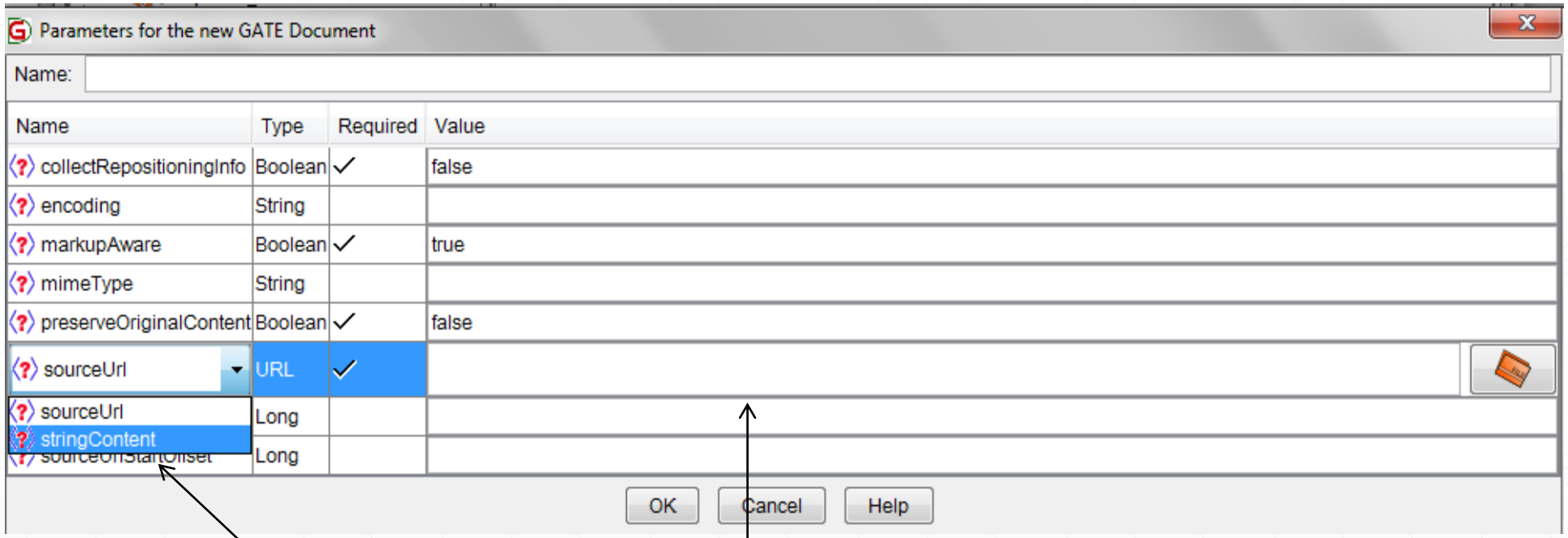
Document Initialisation parameters



Give the document a unique name, or leave it empty as GATE will do that automatically!

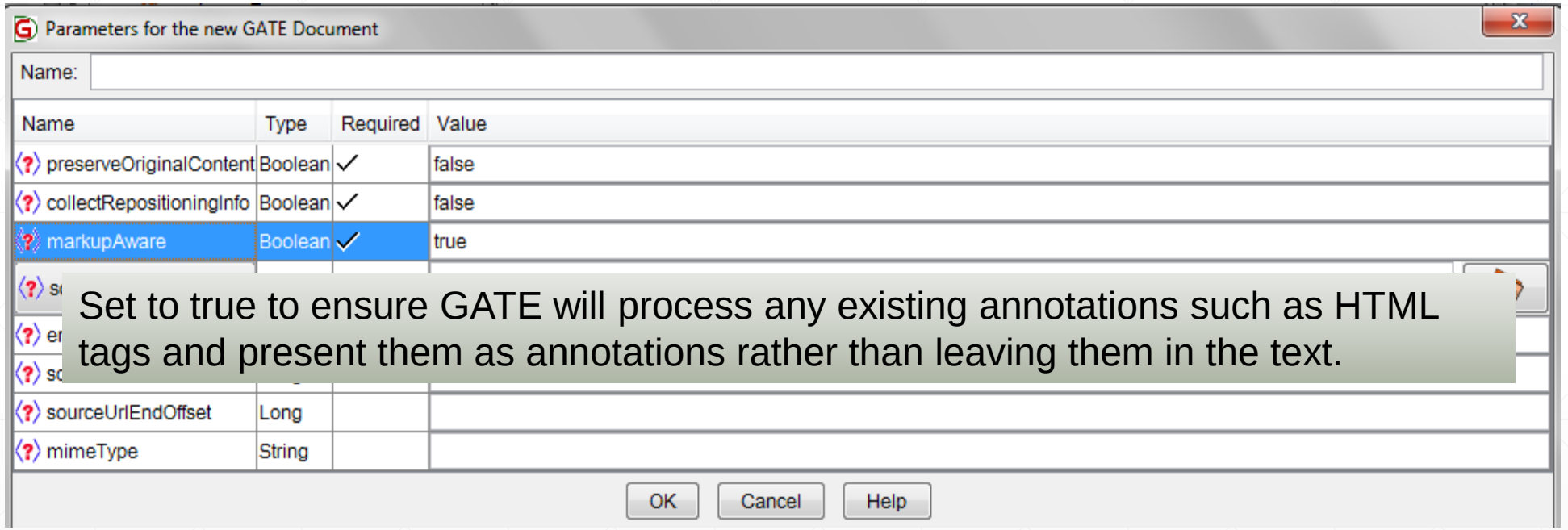
The sourceURL parameter enables you to specify the document to be loaded.
You
can type the filename or URL,
or
click the file browser icon to navigate to the correct document on your PC.

Document Initialisation parameters

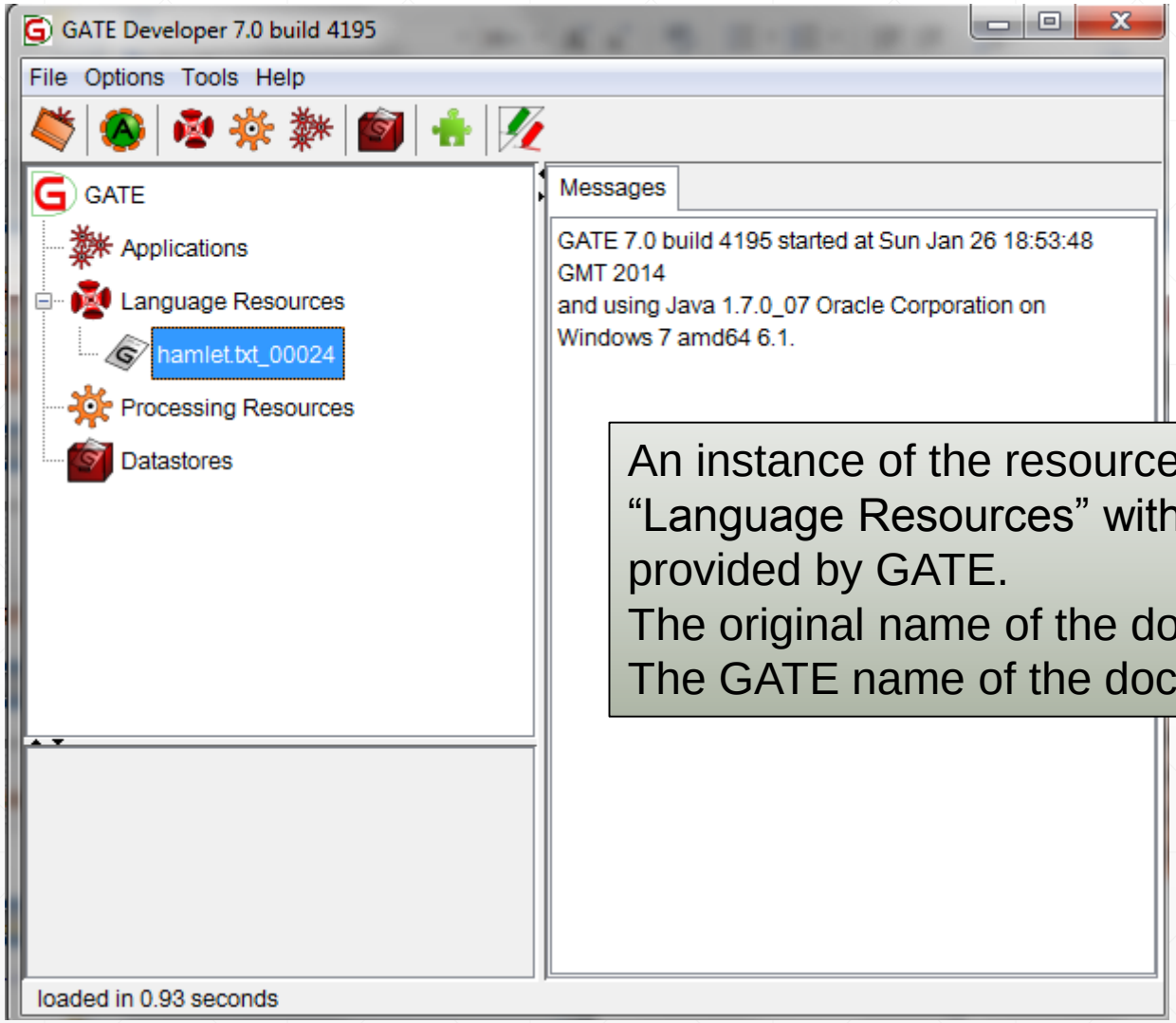


You can also just type a string of text into the box by selecting **stringContent** rather than **sourceUrl**.

Document Initialisation parameters

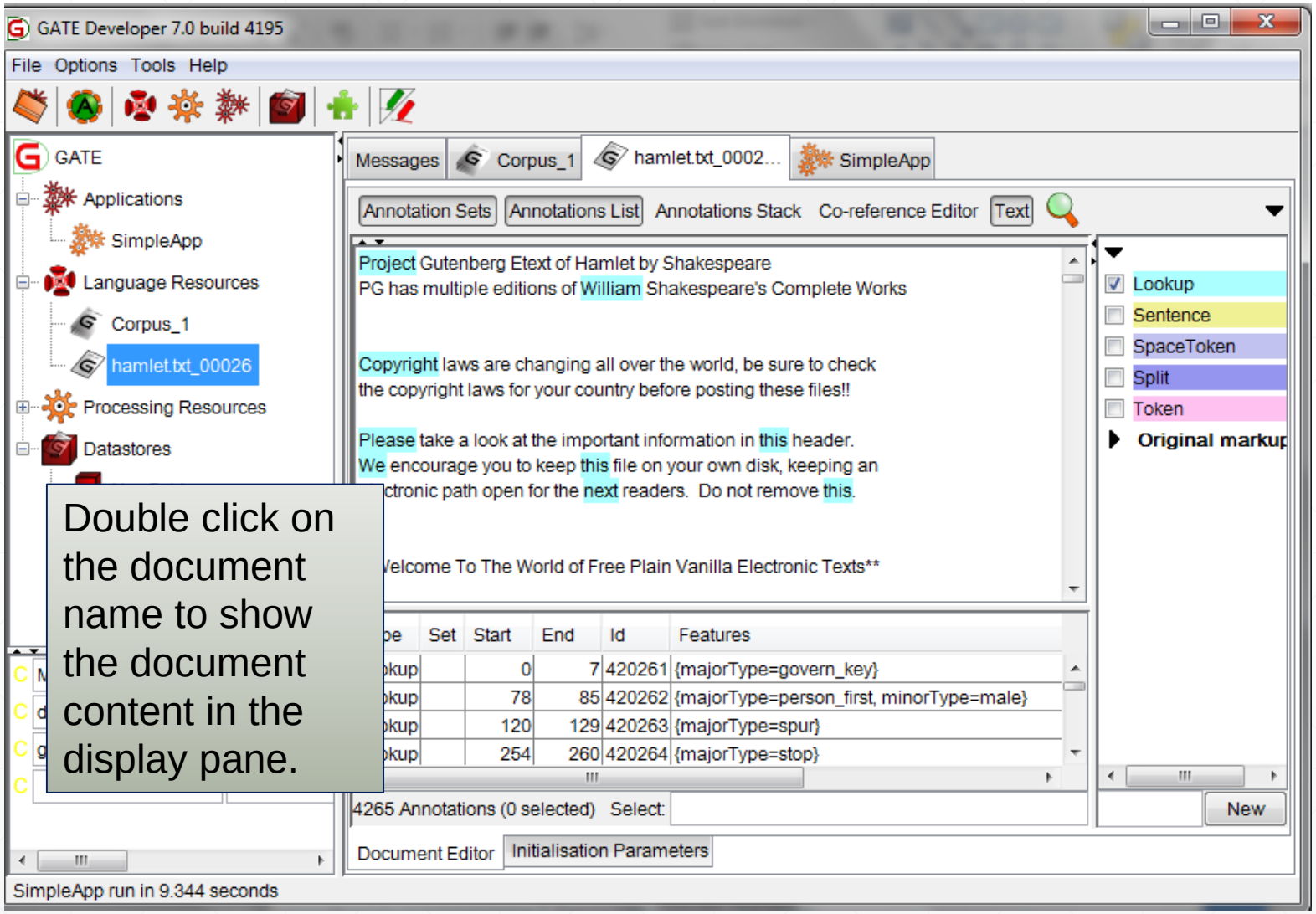


Loading Document in GATE



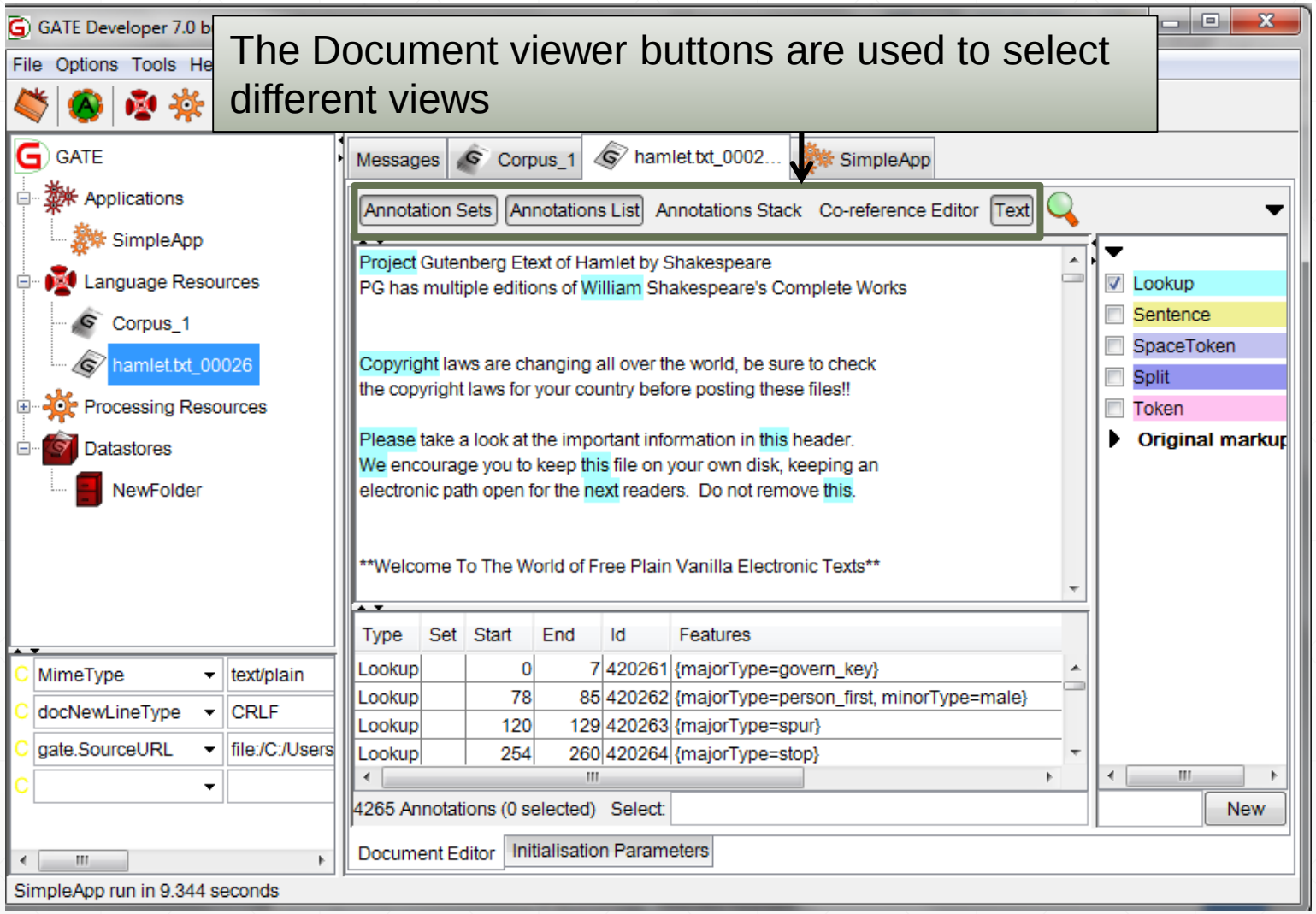
An instance of the resource will appear under “Language Resources” with a unique name provided by GATE.
The original name of the doc is: hamlet.txt
The GATE name of the doc is: hamlet.txt_00024

Viewing the document

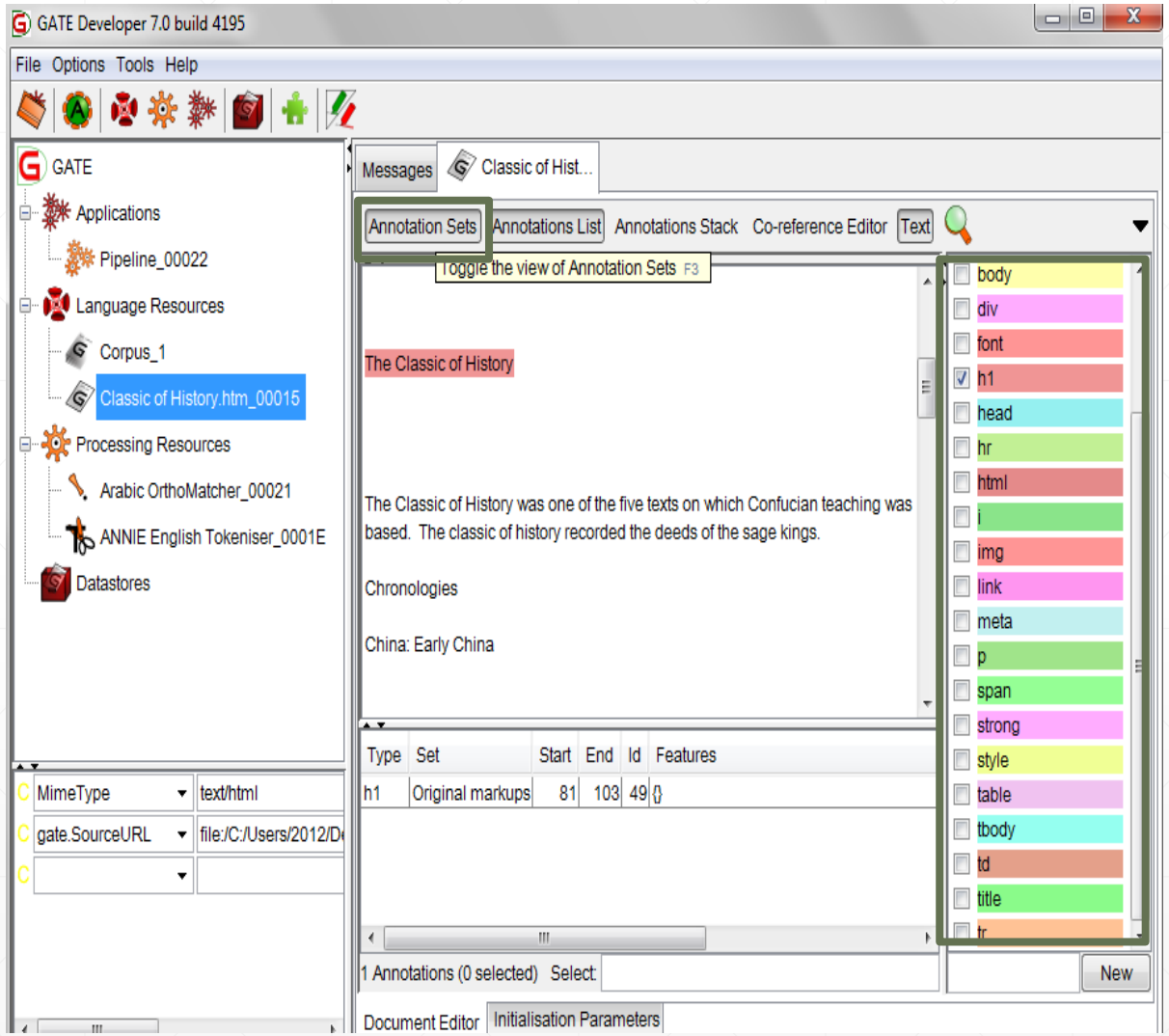


Double click on the document name to show the document content in the display pane.

Viewing the document

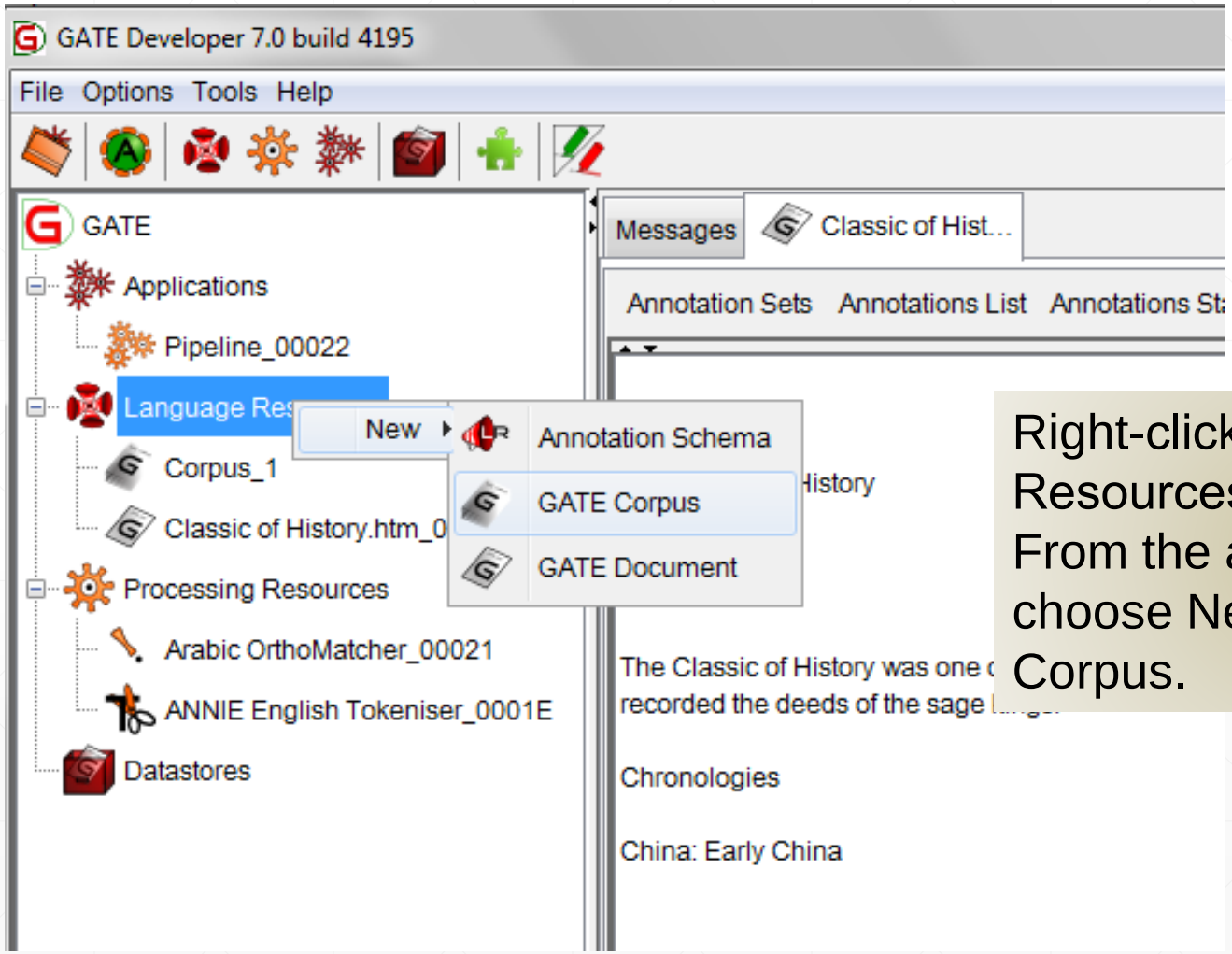


Viewing the document



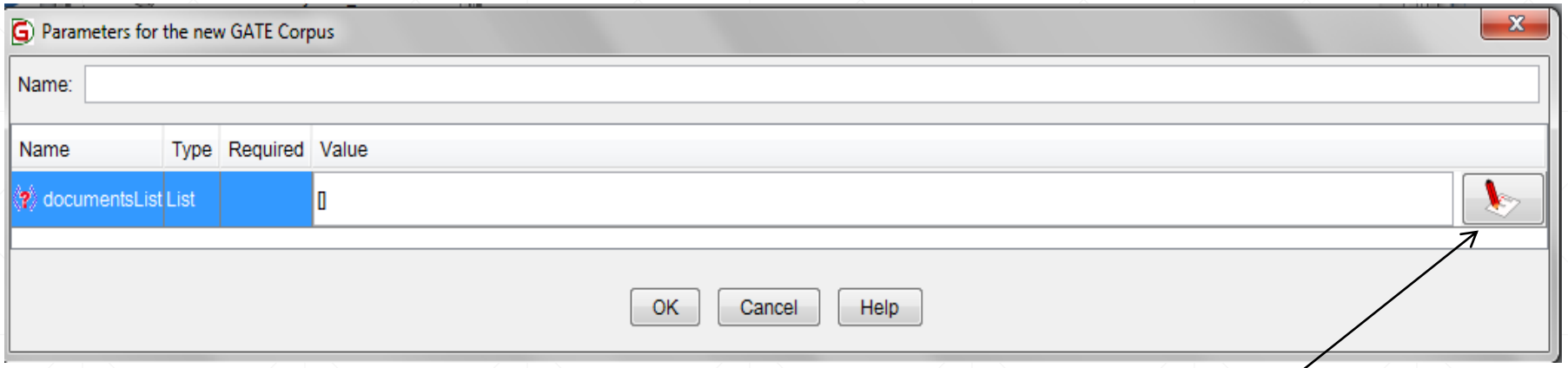
To view the annotations, you need to:
click ‘Annotation Sets’ then select annotation(s) on the right

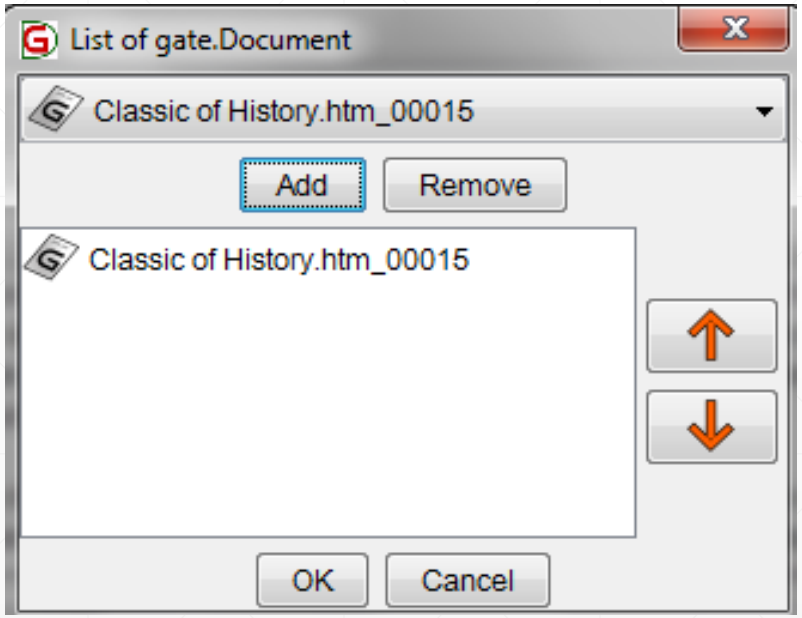
Creating a corpus



Right-click on ‘Language Resources’.
From the appearing menu, choose New->GATE Corpus.

Corpus Initialisation Parameters

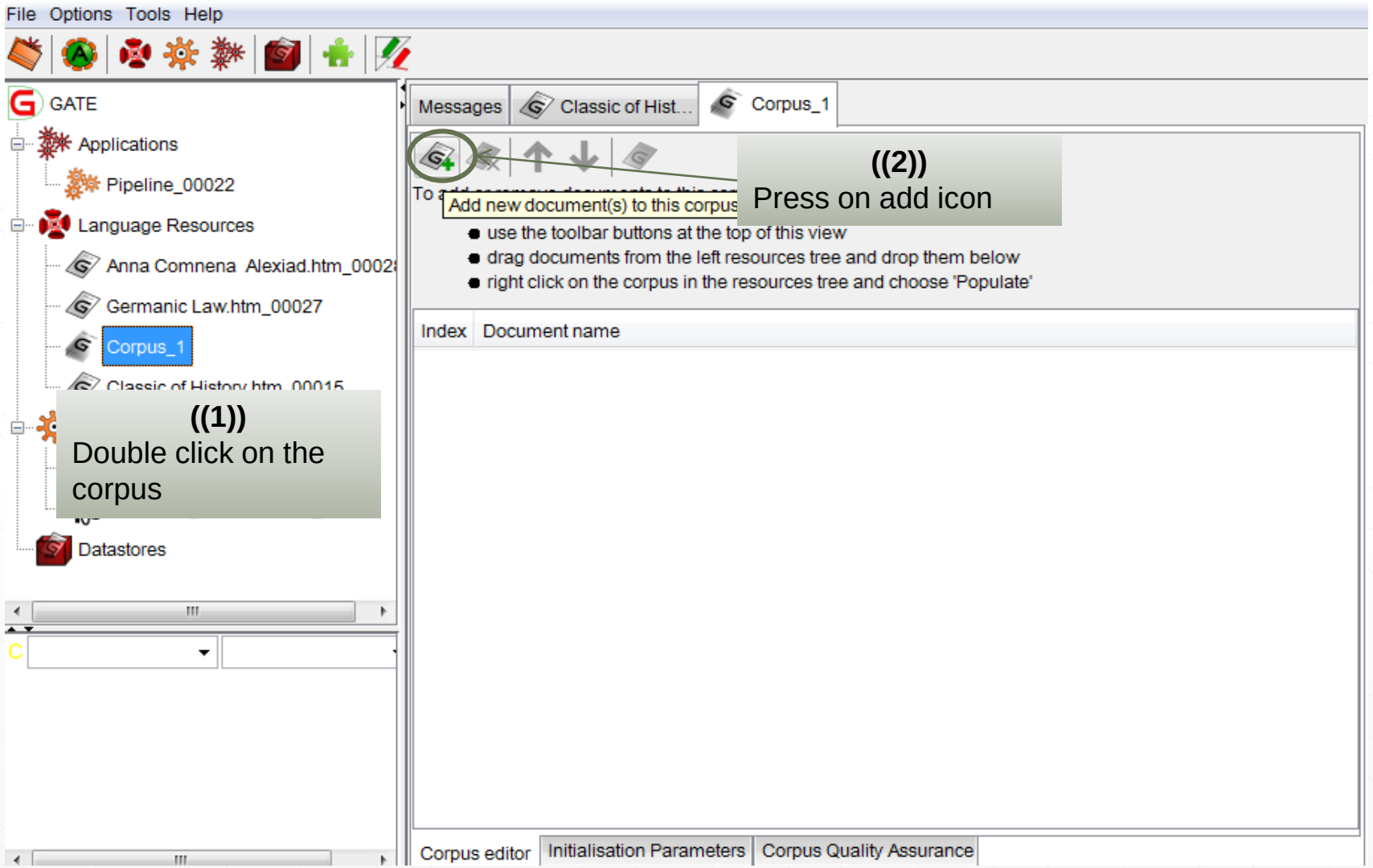




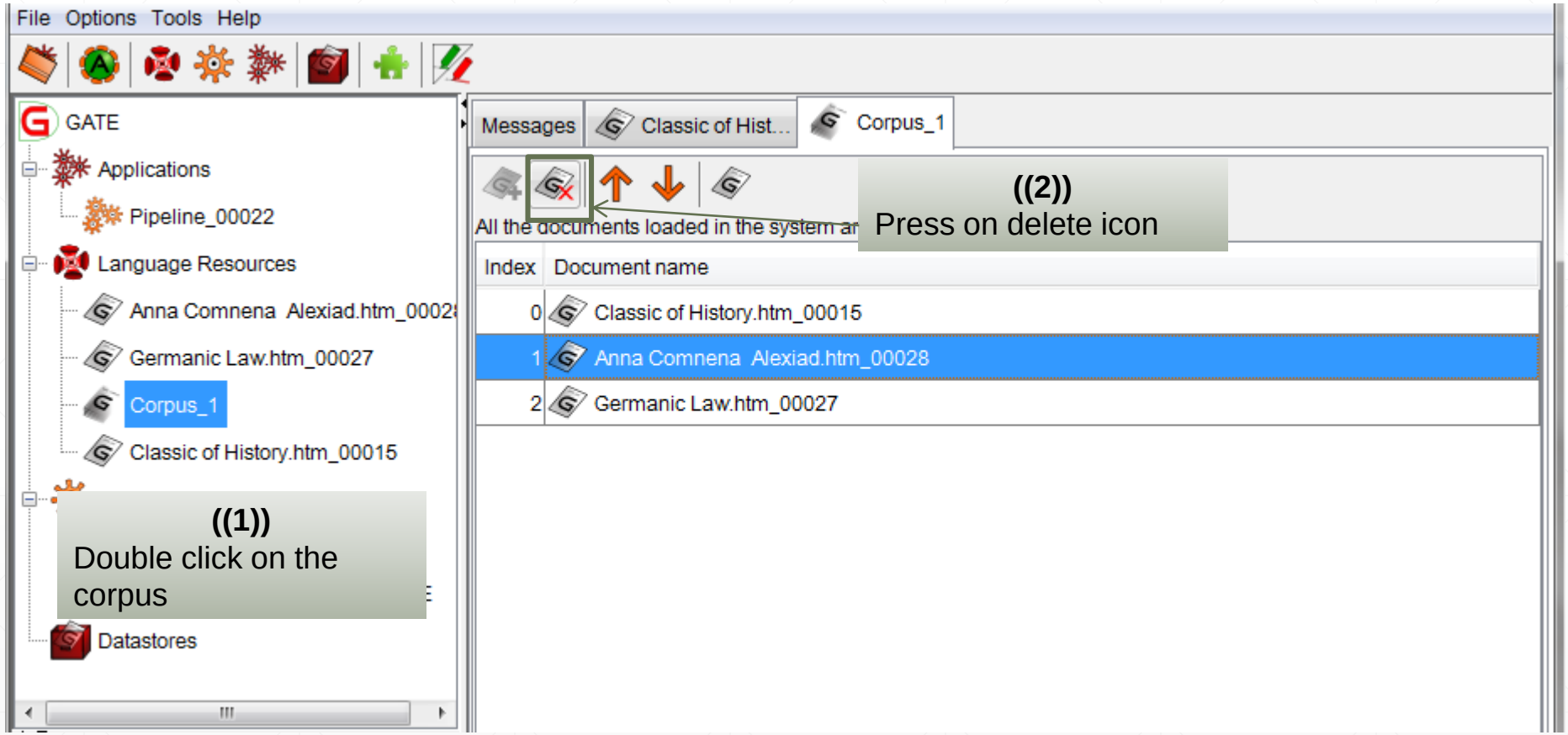
After giving a name to the corpus, click the edit button [add icon] and add documents to the corpus, then press OK.

Pressing OK without adding documents leads to an empty corpus.

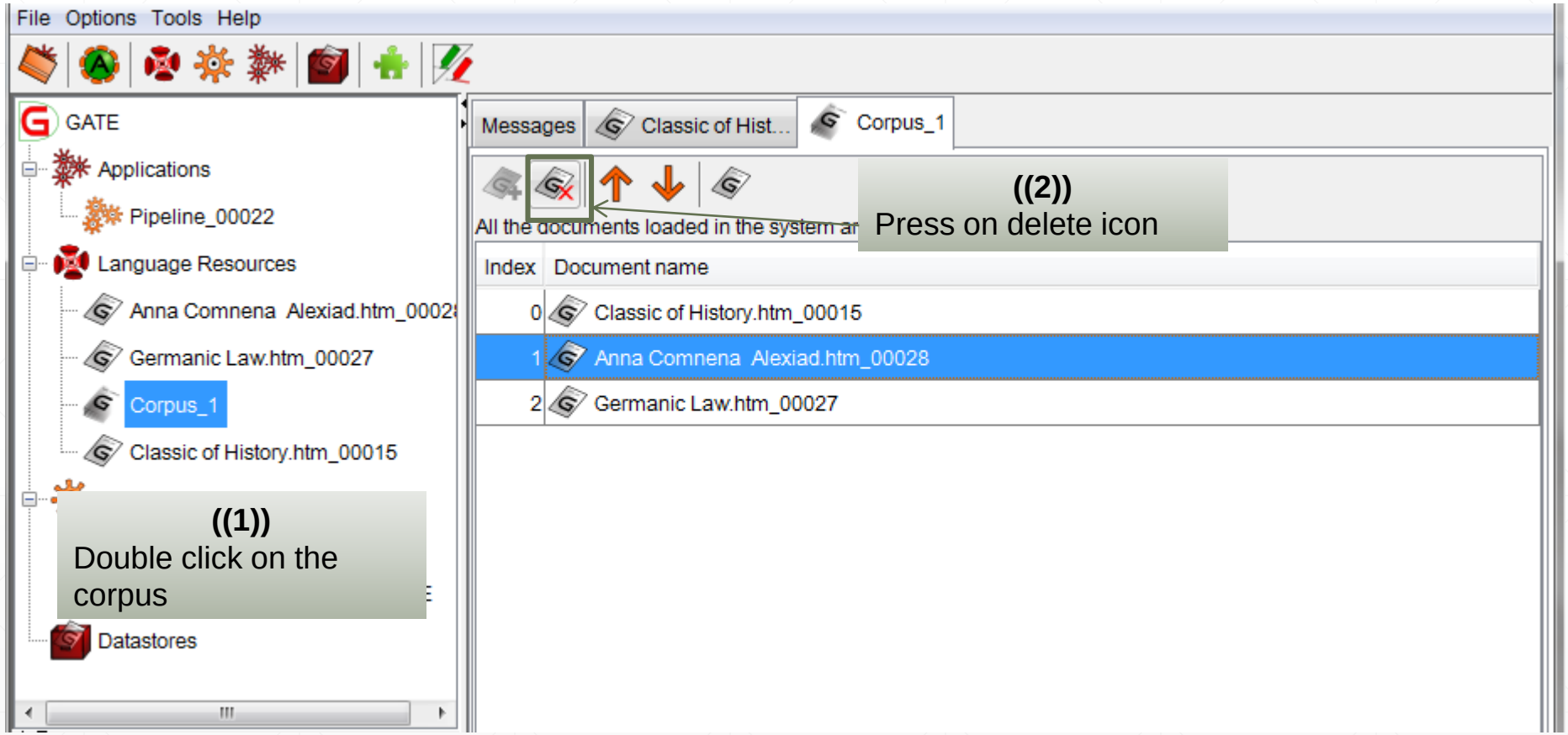
Another way to add documents to a corpus



Removing documents from a corpus



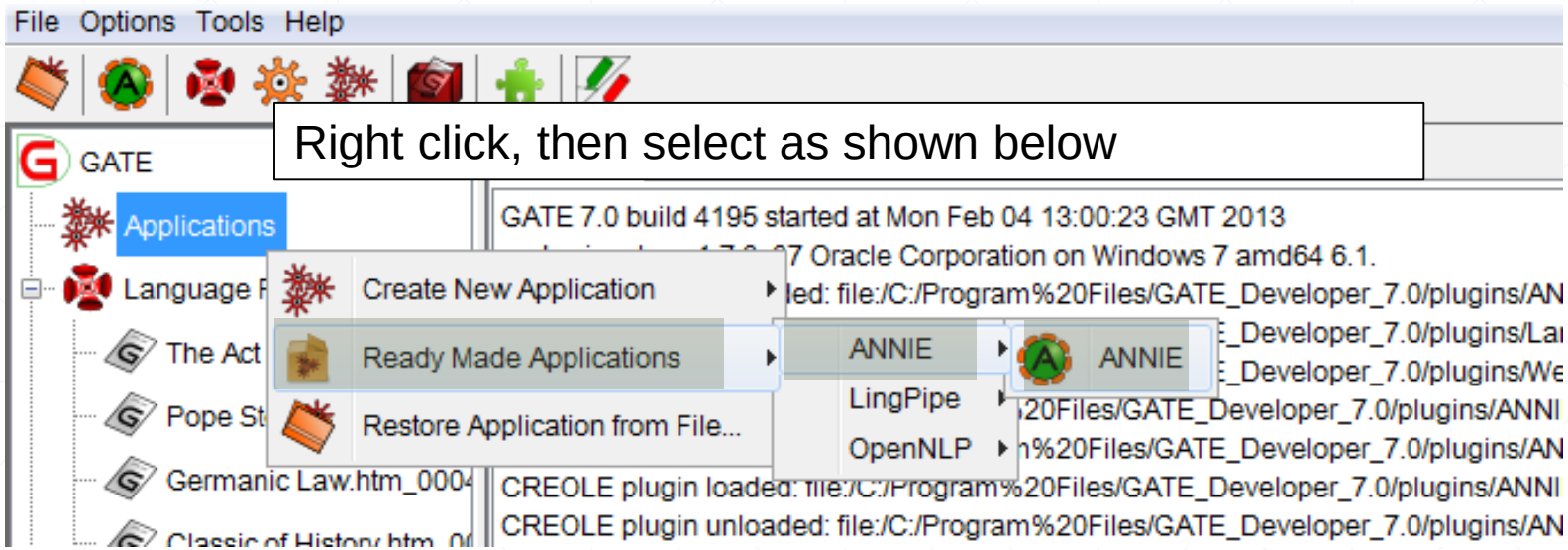
Removing documents from a corpus



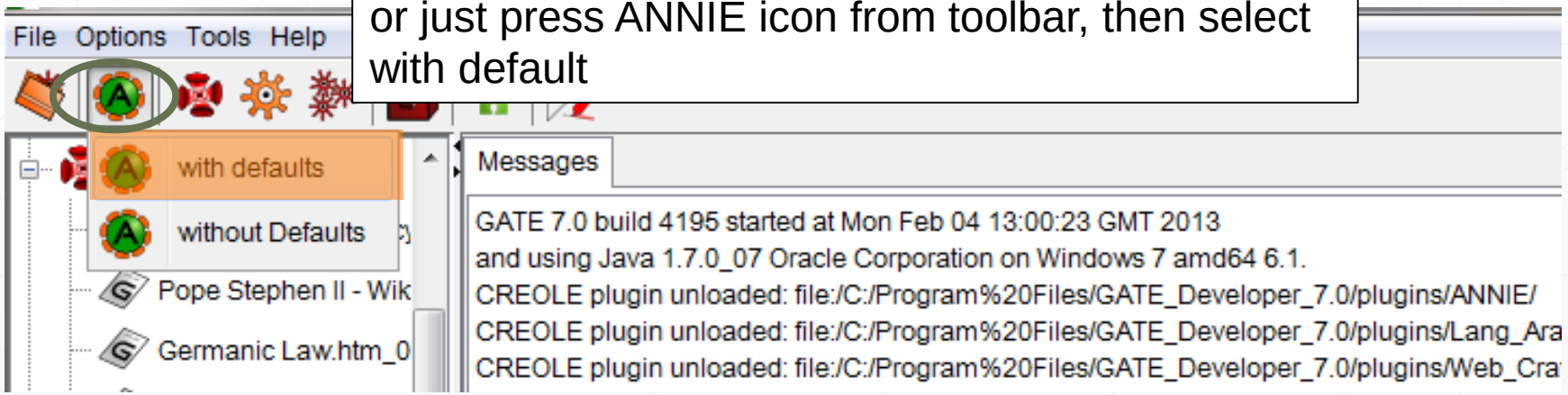
Try…

- Open GATE
- Download the “GATE-hands-on-materials.zip” file from https://sites.google.com/site/mahajalthobaiti/materials
- Load the document “Anna Comnena Alexiad.htm” from “hands-on-materials” folder.
  - Right click on Language Resources and select “New → GATE Document” or
  - File menu → New Language Resource → GATE Document
- A dialogue box will appear.
- Leave the name input box empty.
- Click the file browser icon to navigate to the correct document.
- To view a document, double click on the document name in the Resources pane
- To view the annotations, you first need click “Annotation Sets”, and then select the relevant set and annotation(s) on the right hand side of the GUI
- To see a list of annotations at the bottom, click on “Annotations List”
- Repeat the same processes to load the document “hamlet.txt” from “hands-on-materials”.

Loading and Running ANNIE

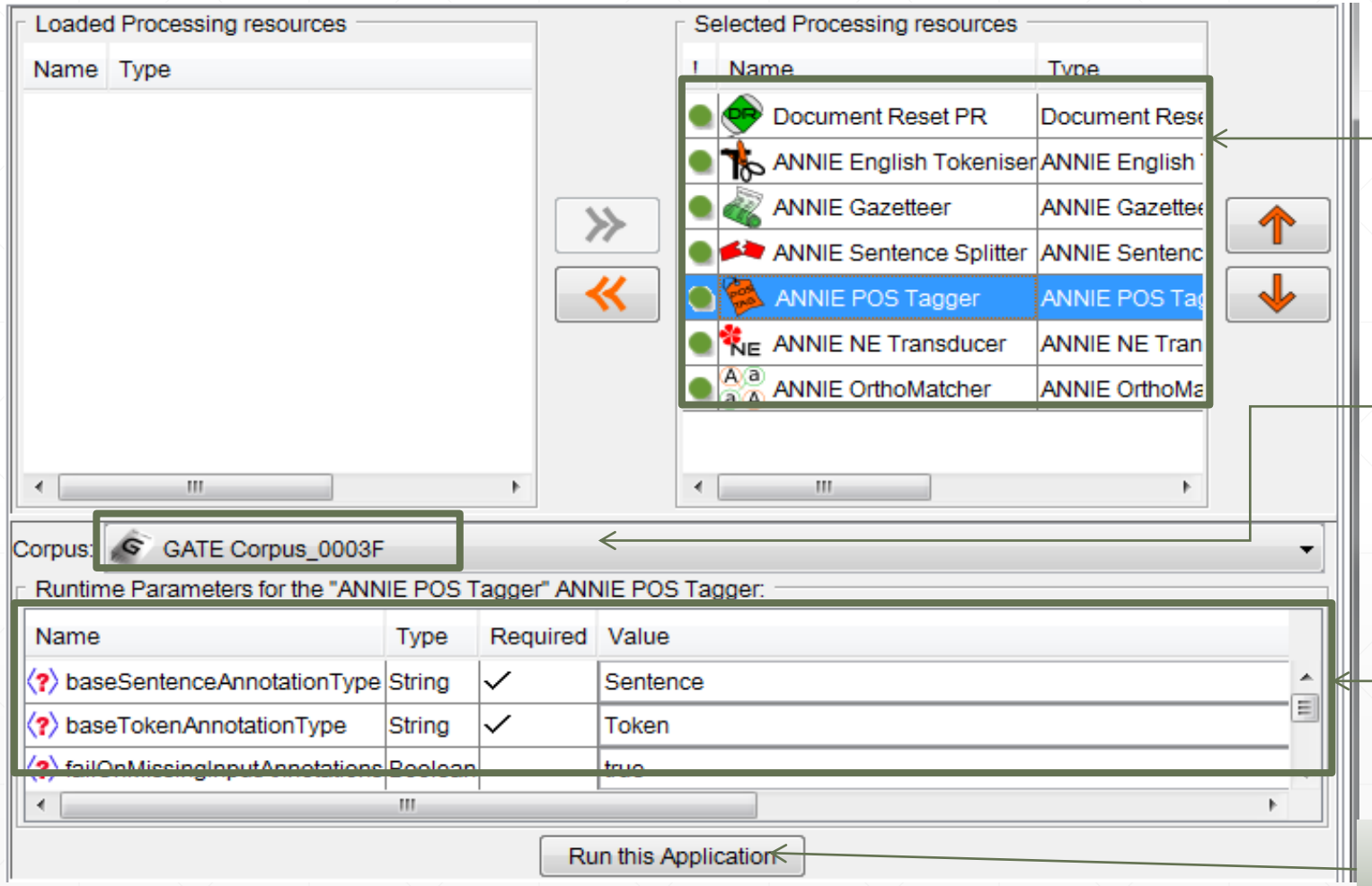


Right click, then select as shown below



or just press ANNIE icon from toolbar, then select with default

Loading and Running ANNIE



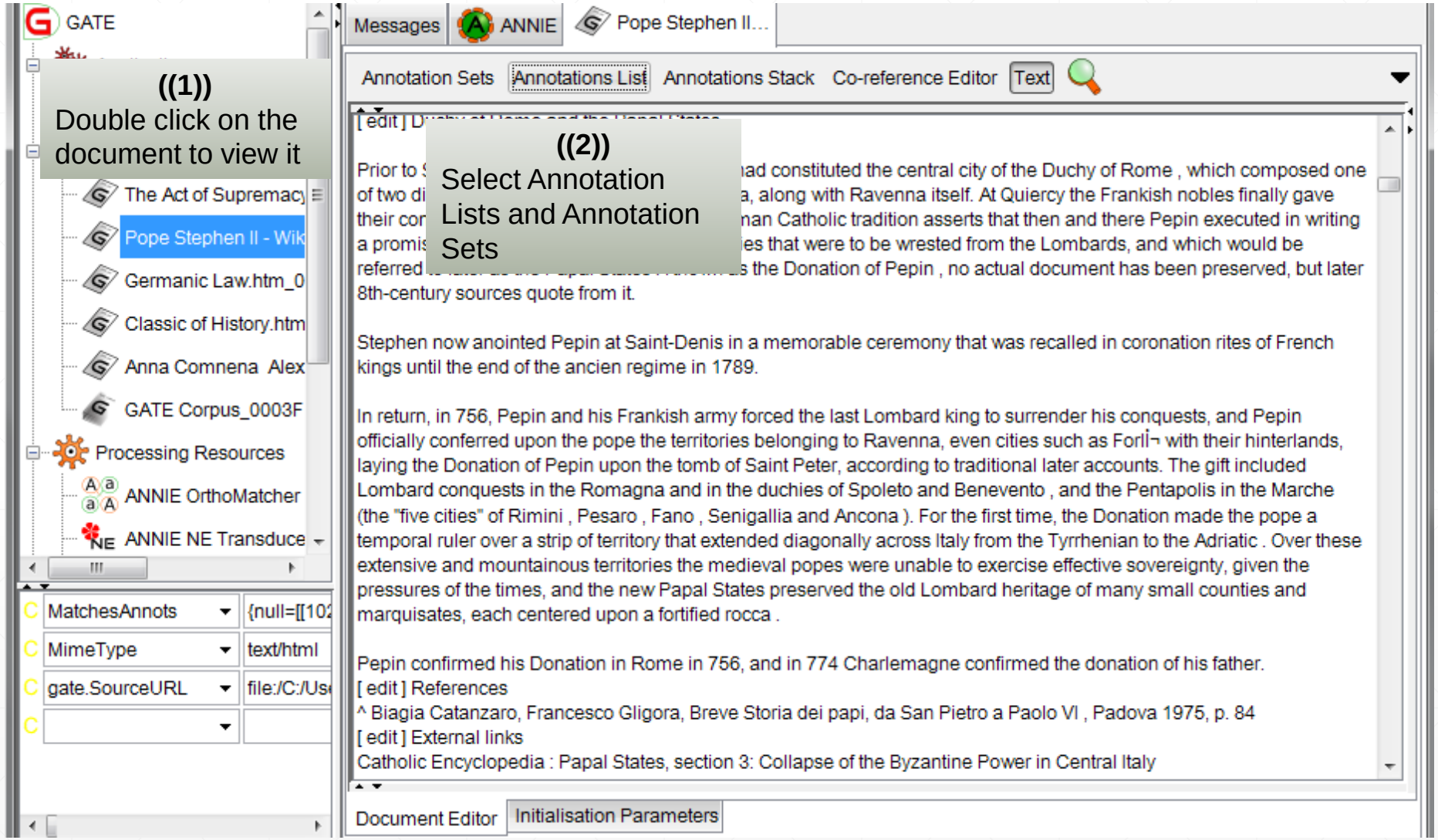
PRs in order of their execution

Corpus on which the application is executed

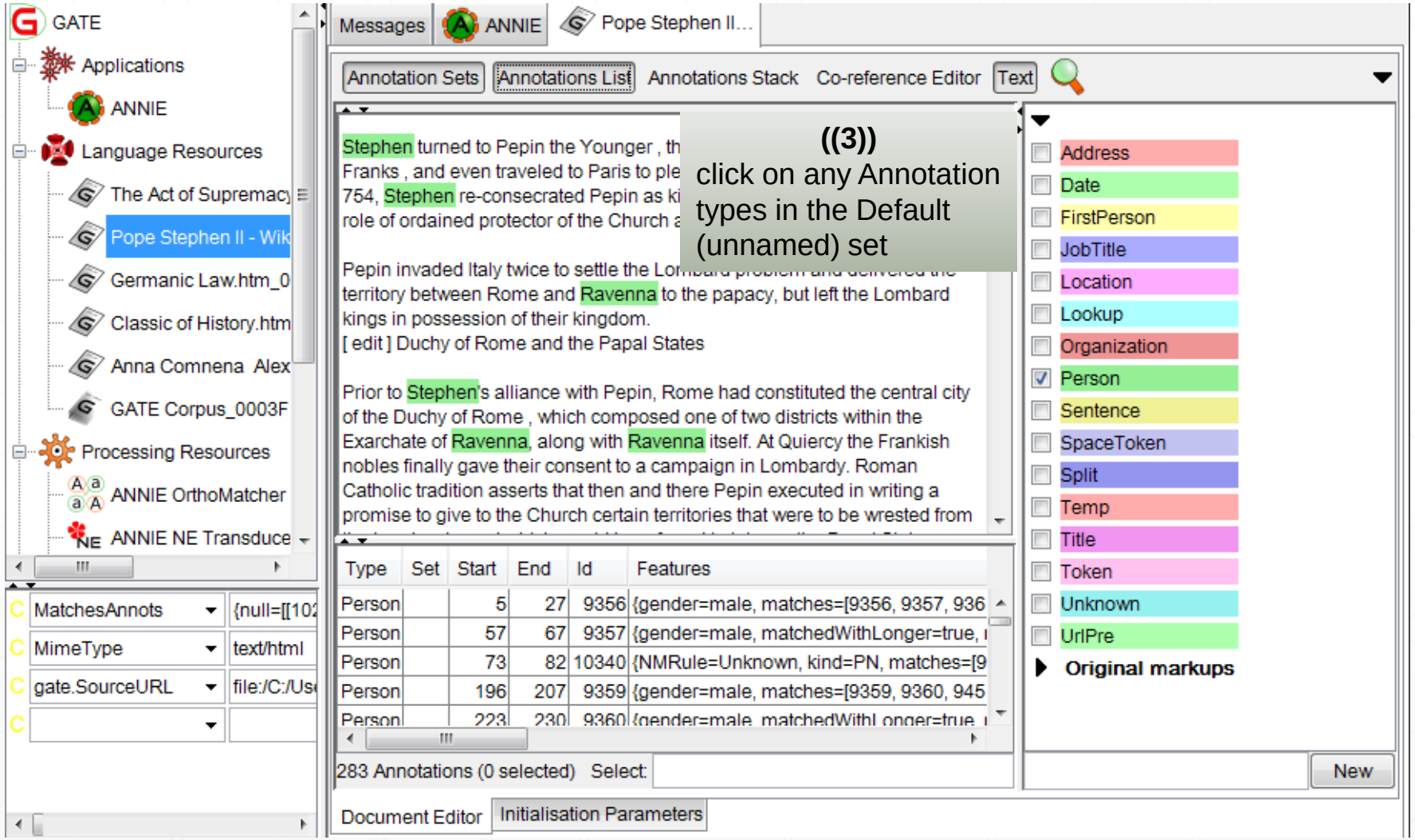
Runtime parameters of the selected PR

Execute the application

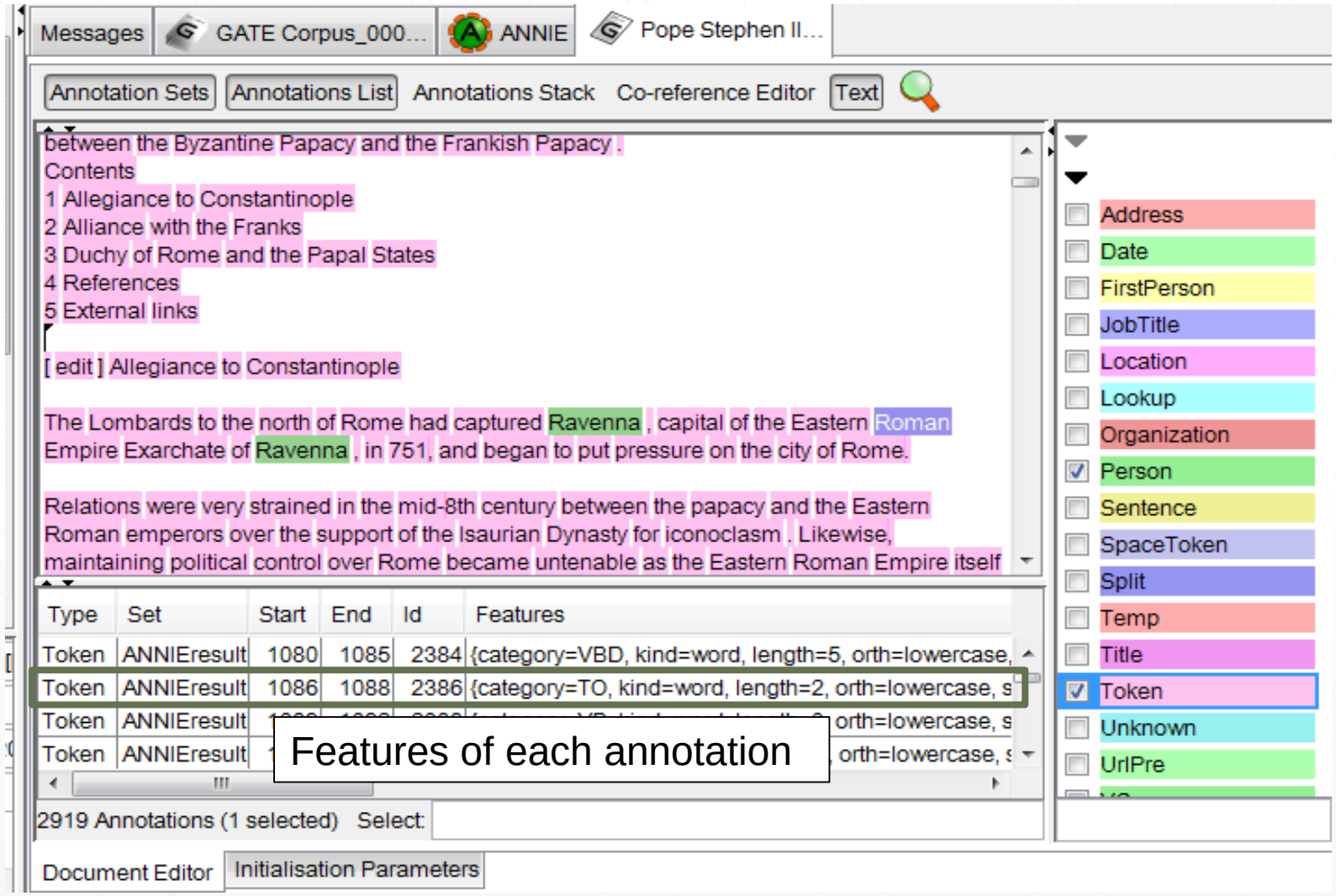
Viewing the Results



Viewing the Results



Viewing the Results



Building Specialized Classifiers

Building Specialized Classifiers

- Disadvantages of ready-made (pre-trained) models:
  - poor performance on domains different from the ones used in building them.
  - The set of NE types can not be changed or extended.
- Solution: building specialized models based on the domain and the NE types of that domain.

The process requires 4 main steps:

- Prepare training and testing datasets.
- Train statistical model
- Test the trained model
- Use the model to perform classification task

Preparing Training & Testing Datasets

- The training and testing sets should be collected from the target domain and annotated manually by experts.
- Training set is used to build and train the models
- Testing set is used to evaluate the trained models
- There are many annotation frameworks that can be followed but the common ones are:
  - ACE
  - CoNLL
  - MUC

Train and Test the Classifier

- Many machine learning algorithms proved to be useful in building and training classifiers:
  - Decision Trees (Weka)
  - CRF (CRF++/CRFsuite)
  - SVM (LibSVM)
  - Naive Bayes (Weka)
  - Etc.

Building specialized Classifiers - Example

- In the following example, we will build a classifier that can recognize three named entities in University domain :
  - person names,
  - course codes, and
  - room numbers.

Training & Testing Data

- University of Essex Corpus (UEC)
  Collected from the documents of the University of Essex domain in 2011.
  available at https://sites.google.com/site/mahajalthobaiti/materials
  (The training and test sets used in the Lab are only parts of the whole UEC).
- CoNLL Framework:
  - A token for each line
  - columns separated by a single space
    <Token feature1 feature2 … annotation>
  - An empty line after each sentence
  - Tags in IOB format
    B-PER: The Beginning of the name of a person.
    I-PER: The continuation (Inside) of the name of a person.
    B-COR: The Beginning of a course number.
    I-COR: The continuation (Inside) of a course number.
    B-ROM: The Beginning of a room number.
    I-ROM: The continuation (Inside) of a room number.

Training & Testing Data

Dean of the Graduate School Dr Pam Cox said: “This is a great result for Essex.”

```
Dean        O
of          O
the         O
Graduate    O
School      O
Dr          O
Pam         B-PER
Cox         I-PER
said        O
:           O
"           O
This        O
is          O
a           O
great       O
result      O
for         O
Essex       O
.           O
```

Extracting features

Features should be extracted from text to be used in the training data. For example:

- F1= The current word (w_i) is capitalized.
- F2= The previous word (w_{i-1}) is capitalized.
- F3= The next word (w_{i+1}) is capitalized.
- F4= The word (w_{i-1}) that appears before the current word.
- F5= The word (w_{i+1}) that appears after the current word.

Sample of Final Dataset
(Dataset with Extracted Features)

	F1	F2	F3	F4	F5	Annotation
and …						
Dr	1	0	1	and	Gary	O
Gary	1	1	1	Dr	Armstrong	B-PER
Armstrong	1	1	0	Gary	at	I-PER
at ……						

Sample of Final Dataset
(Dataset with Extracted Features)

Dean	O
of	O
the	O
Graduate	O
School	O
Dr	O
Pam	B-PER
Cox	I-PER
said	O
:	O
"	O
This	O
is	O
a	O
great	O
result	O
for	O
Essex	O
.	O

Dean	1	0	0	null	of	O
of	0	1	0	dean	the	O
the	0	0	1	of	graduate	O
Graduate	1	0	1	the	school	O
School	1	1	1	graduate	dr	O
Dr	1	1	1	school	pam	O
Pam	1	1	1	dr	cox	B-PER
Cox	1	1	0	pam	said	I-PER
said	0	1	0	cox	:	O
:	0	0	0	said	"	O
"	0	0	1	:	this	O
This	1	0	0	"	is	O
is	0	1	0	this	a	O
a	0	0	0	is	great	O
great	0	0	0	a	result	O
result	0	0	0	great	for	O
for	0	0	1	result	essex	O
Essex	1	0	0	for	.	O
.	0	1	0	essex	null	O

Write a code that generates the above features for the University dataset !

CRF++

Available as an open source software

https://taku910.github.io/crfpp/

- Download the Binary package for MS-Windows.
- Unzip the package.
- Keep training and testing datasets in the distribution directory.
- Prepare the CRF++ template for the features.

CRF++

The ‘template’ is a way to represent the **features in crf++ toolkit**

Example of representing template in CRF++ [1] .

Training dataset

He	$< f_{11} >$	$< f_{12} >$	
Reckons	$< f_{21} >$	$< f_{22} >$	
the	$< f_{31} >$	$< f_{32} >$	<< CURRENT TOKEN
current	$< f_{41} >$	$< f_{42} >$	
account	$< f_{51} >$	$< f_{52} >$	

Template	Expanded feature
%x[0,0]	the
%x[0,1]	$< f_{31} >$
%x[-1,0]	reckons
%x[-2,1]	$< f_{11} >$
%x[0,0]/%x[0,1]	the/f31

[1] https://taku910.github.io/crfpp

CRF++

- To train Unimodel on the UniTrainingSet, use the following command inside the distribution directory:

```
>> crf_learn template UniTrainingSet Unimodel
```

Training parameters:

- -a CRF-L2 or CRF-L1: Changing the regularization algorithm. Default setting is L2.
- -c float: With this option, you can change the hyper-parameter for the CRFs. With larger C value, CRF tends to overfit to the give training corpus. This parameter trades the balance between overfitting and underfitting.
- -f NUM: This parameter sets the cut-off threshold for the features. CRF++ uses the features that occurs no less than NUM times in the given training data. The default value is 1.
- -p NUM: If the PC has multiple CPUs, you can make the training faster by using multi-threading. NUM is the number of threads.

Example:

```
>> crf_learn -f 3 -c 1.5 template UniTrainingSet Unimodel
```

Evaluating CRF's predictions

- To evaluate the trained model, use the command:

```
>> crf_test –m Unimodel UniTestingSet > testResult
```

- The result file 'testResult' will contain a column for the estimated tags by the trained model.
  - Write a code to compute the Precision, Recall, F-measure from the 'testResult' file.

  OR

  - Use the evaluation software used in the CoNLL shared task http://www.cnts.ua.ac.be/conll2002/ner/bin/conlleval.txt

    After downloading, rename the file from 'conlleval.txt' to 'conlleval.pl'.

    It is written in Perl. So, it requires Perl to be installed on your machine.